UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Cursos partilhados no sistema Fenix
- Análise comparativa de alternativas de concretização -

Diogo José Nunes Godinho

Mestrado em Engenharia Informática
Especialização em Engenharia de Software

Trabalho de Projeto orientado por:
Prof.ª Doutora Maria Dulce Pedroso Domingos
Prof. Doutor Carlos Nuno da Cruz Ribeiro

2020

Agradecimentos

Em primeiro lugar, queria agradecer à Reitoria da Universidade de Lisboa por me ter dado a possibilidade de realizar a minha tese de mestrado. Aos meus orientadores Professora Dulce Domingos e Professor Carlos Ribeiro.

A todo o Departamento de Informática da Reitoria pela disponibilidade ao longo do projeto, nomeadamente à Ana Rute, Andreia Rainha, Catarina Silva, Daniel Vitoriano, Daniela Mendes, Fábio Ferreira, José Lima, Nuno Heitor, Nuno Jesus, Pedro Moita, Ricardo Rito e Tânia Crespo.

Aos colegas bolseiros que fizeram parte desta jornada, Ana Espinheira, Diana Marques, Francisco Caeiro, Mafalda Luz e Pedro Ladino.

Ao Agrupamento 1260 - Bela Vista, especialmente à Patrulha Corvo, Bando Amarelo e claro à Alcateia 138.

Aos meus amigos, especialmente à Mara Santos e Miriam Leal.

Por último mas não menos importante à minha família, pela paciência e por me aturarem ao longo deste anos.

Obrigado a todos.

Dedicatória.

”Eu sinto que o universo conspira para me fazer feliz.”

Ricardo Araújo Pereira

Resumo

Os cursos partilhados da Universidade de Lisboa (ULisboa) são cursos que são lecionados em conjunto por duas ou mais escolas da Universidade, podendo apresentar três tipologias: associação, gestão rotativa ou co-tutela. No caso de cursos em associação, os alunos realizam o tronco comum numa escola e realizam Unidades Curriculares (UCs) noutras, inscrevendo-se em UCs isoladas, ou em *majors*/*minors* nas outras escolas. Nos cursos em gestão rotativa existem várias escolas que gerem o curso num sistema de rotatividade, isto é, num ano letivo o curso é gerido por uma escola, no ano seguinte é gerido por outra escola. Nos cursos que são em co-tutela, cada semestre ou ano curricular é lecionado em escolas diferentes, por exemplo o primeiro e terceiro anos curriculares são realizados numa escola e o segundo é realizado noutra.

Atualmente, o Sistema Integrado de Gestão Académica da ULisboa, designado por Fenix, suporta a gestão dos serviços académicos, do planeamento e recursos e da tesouraria das escolas. Apesar das várias escolas utilizarem o mesmo sistema, estas instâncias não comunicam entre si. Deste modo, várias tarefas de suporte são realizadas manualmente e normalmente através de troca de mensagens de correio eletrónico entre os serviços académicos de cada uma das escolas. Isto leva a um maior esforço por parte dos serviços e pode potenciar a ocorrência de erros.

Tendo como objetivo solucionar-se este problema, que passa por tornar as instâncias comunicantes entre si, foi realizada uma análise mais pormenorizada do que envolve suportar a gestão de cursos partilhados no Sistema FenixEdu (Fenix). Foram estudadas várias alternativas e chegou-se a uma implementação que serve como prova de conceito, permitindo concluir que é possível suportar as funcionalidades subjacentes à gestão de cursos partilhados sem grandes impactos no desempenho atual do sistema.

Palavras-chave: Sistema Fenix, Cursos partilhados, Sistemas distribuídos

Abstract

The shared courses of ULisboa are courses that are taught together by two or more schools of the University, and may present three typologies: association, rotating management or co-guardianship. In the case of association courses, students perform the common branch in one school and carry out Curricular Units (CUs) in others, enrolled in isolated CUs, or majors/minors in other schools. In rotating management courses there are several schools that manage the course in a rotating system, i.e. in one academic year, the course is managed by a school, in the following year it is managed by another school. In co-guardianship courses, each semester or curricular year is taught in different schools, for example, the first and third curricular years are held in one school and the second is held in another.

Currently, the Integrated Academic Management System of ULisboa, called Fenix, supports academic services management, planning and resources management and the schools' treasury management. Although several schools use the same system, these instances do not communicate with each other. In this way, various support tasks are performed manually and usually by exchanging emails between the academic services of schools. This leads to increased effort on the part of services and can enhance the occurrence of errors.

With the aim of solving this problem, which involves making the instances communicate with each other, a more detailed analysis of what involves supporting the shared courses' management in Fenix has been carried out. Several alternatives have been studied and an implementation has been reached that serves as proof of concept, enabling to conclude that Fenix can support the underlying shared course management features without major impacts on the current system performance.

Keywords: Fenix System, Shared Courses, Distributed Systems

Conteúdo

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Capítulo 1

Introdução

O projeto apresentado analisa a problemática dos cursos partilhados no sistema Fenix da ULisboa, desenvolvido no âmbito da disciplina de Dissertação/Projeto de Engenharia Informática do Mestrado em Engenharia Informática da Faculdade de Ciências da ULisboa. Este capítulo descreve a motivação, os objetivos e as contribuições do projeto, assim como a estrutura do documento.

1.1 Motivação

No ano letivo de 2020/2021, todas as escolas da Universidade de Lisboa utilizarão o Sistema Integrado de Gestão Académica (SIGA), Fenix. No entanto, os sistemas Fenix de cada escola são autónomos e não interagem.

Os cursos partilhados da ULisboa são cursos que são lecionados por duas ou mais escolas da Universidade. Estes cursos podem ter três tipologias: associação, co-tutela e gestão rotativa. Nos cursos em associação, o tronco comum é lecionado numa escola e as restantes UCs podem ser lecionadas noutras escolas como UCs isoladas ou em *majors/-minors*. Nos cursos em co-tutela, a gestão do curso é partilhada, isto é, cada semestre ou ano curricular é lecionado numa das escolas. Por último existem os cursos de gestão rotativa, onde num ano o curso é gerido por uma escola e no ano seguinte é gerido pela outra escola.

A gestão dos cursos partilhados requer a transferência de informação sobre os cursos, os alunos, as turmas e as notas, de uma das escolas para a outra, por vários meios, seja via correio eletrónico ou via telefone. Estes processos originam uma comunicação exaustiva entre os serviços académicos das escolas, o que leva a um elevado esforço das pessoas e uma grande demora na comunicação, que pode atrasar outros processos das escolas. Pode também levar à introdução de erros humanos nas informações. Justifica-se deste modo a necessidade de automatizar e uniformizar os processos, reduzindo o seu tempo e o risco de erros de comunicação.

1.2 Objetivos

O objetivo deste projeto é realizar uma análise comparativa de alternativas de concretização que visam o suporte da gestão de cursos partilhados entre as várias instâncias do Fenix.

O projeto foi dividido em três fases com objetivos específicos:

Análise do problema

Os objetivos desta fase são fazer uma análise ao domínio e âmbito do problema, realizando o levantamento e a análise de requisitos junto de algumas escolas onde existem cursos partilhados.

Desenho da solução

Tendo por base os resultados da fase anterior, bem como as especificações do Fenix, o principal objetivo desta fase é estudar alternativas de arquitetura e desenho que permitam estender o sistema Fenix com as funcionalidades de gestão dos cursos partilhados.

Validação da solução

Esta última fase tem como objetivo a validação da solução implementada, dando particular relevo ao seu desempenho.

1.3 Contribuições

As principais contribuições deste trabalho são:

Os casos de uso da análise de requisitos

Foi realizado o levantamento de requisitos em conjunto com o Departamento de Informática dos Serviços Centrais (DI SC) e os serviços académicos, da Faculdade de Letras da ULisboa e do Instituto de Geografia e Ordenamento do Território. Este levantamento de requisitos permitiu conhecer com mais detalhe a forma de funcionamento dos cursos partilhados na ULisboa e definir os principais casos de uso.

A proposta da arquitetura e desenho da solução

Foram analisadas diversas alternativas no que diz respeito aos estilos arquiteturais, às arquiteturas de sistemas, à gestão da coerência dos dados e modelação do domínio da solução. Destas alternativas foi criada uma proposta para a arquitetura e desenho da solução.

A prova de conceito

Foi implementada uma prova de conceito de forma a validar a possibilidade das funcionalidades subjacentes à gestão de cursos partilhados poderem ser concretizadas no Fenix, utilizando a proposta da arquitetura e desenho escolhidos. Os testes de desempenho mostram que a solução é exequível.

1.4 Estrutura do documento

O documento tem a seguinte estrutura:

Capítulo 1 - Introdução
Apresenta a motivação, os objetivos, as contribuições e a estrutura do documento.

Capítulo 2 - Contexto: Sistema Fenix
Descreve o sistema Fenix, nomeadamente as suas principais tecnologias, linguagens de modelação e arquitetura.

Capítulo 3 - Levantamento de requisitos
Apresenta o levantamento de requisitos, através de casos de uso que descrevem a forma como são geridos os cursos partilhados atualmente.

Capítulo 4 - Análise de requisitos
Define os principais casos de uso do sistema, bem como os seus diagramas de sequência.

Capítulo 5 - Arquitetura, desenho e implementação
Analisa as alternativas estudadas do ponto de vista da arquitetura e do desenho da solução e apresenta alguns detalhes de implementação.

Capítulo 6 - Validação
Apresenta e analisa os testes de desempenho.

Capítulo 7 - Conclusão
Conclui o trabalho discutindo os resultados obtidos e o trabalho futuro.

Capítulo 2

Contexto: Sistema Fenix

O Sistema Integrado de Gestão Académica (Fenix) começou a ser desenhado em 2002 no Instituto Superior Técnico (IST). Desde 2015, tem vindo a ser implementado na Universidade de Lisboa (ULisboa), de forma faseada. Atualmente, o sistema Fenix é usado em dezasseis das dezoito escolas.

O sistema Fenix dá suporte à gestão e *back office* académicos, incluindo *workflow* para grande parte dos processos, tais como o desenho, planeamento e aprovação de cursos e unidades curriculares. Fornece também apoio nas tarefas académicas de processos de admissão, inscrições *on-line*, processos de avaliações, notas, entre outras [9].

2.1 Tecnologias

O sistema Fenix utiliza várias tecnologias. As principais, que são utilizadas em todos os projetos, são as seguintes:

Java 11

O Java é uma linguagem de programação. O código Java é compilado para *bytecodes* que correm em *Java Virtual Machine* (JVM), o que permite a mesma aplicação correr em diversos sistemas operativos sem que sejam necessárias alterações ao código fonte. [6].

Maven

O Maven é uma ferramenta que tem como objetivo facilitar o desenvolvimento de projetos em Java, sendo usada para compilar e gerir um projeto, especialmente útil quando se trata de um projeto com um grande número de módulos, facilitando a gestão das dependências [4].

Tomcat 9

O Tomcat é um *web server open source*, que permite a existência de um servidor puramente em java, implementando Java Servlets (classe Java que responde a pedidos HTTP)

e *Java Server Pages (JSP)* (responsável por criar páginas *web* dinâmicas). O servidor trata cada pedido HTTP numa *thread* separada, contudo todos os processos são executados numa mesma JVM [5].

MySQL

O MySQL é um *Relational Database Management System* (RDBMS) *open source*, baseada em SQL. O MySQL utiliza um modelo cliente-servidor onde são efetuados pedidos e são recebidas respostas aos mesmos. Permite também a existência de procedimentos, isto é, programação interna da base de dados [7].

Git

O Git é um sistema distribuído de controlo de versões *open source* que permite manter um histórico de versões ao longo do desenvolvimento de um projeto. Como é distribuído, permite que várias pessoas utilizem o mesmo repositório e contribuam para o desenvolvimento do projeto [2].

2.2 Linguagens das *frameworks* Fenix e *OMNIS*

As *frameworks* Fenix e *OMNIS* utilizam linguagens de modelação de domínio e de apresentação. Estas linguagens facilitam a modelação do projeto, contribuindo para reduzir o tempo de desenvolvimento.

2.2.1 Linguagens de modelação de domínio

As linguagens de modelação de domínio das *frameworks* Fenix e *OMNIS* são a DML e a DSL, respetivamente. A existência destas duas linguagens é justificada por questões históricas e de compatibilidade com versões anteriores. A grande diferença entre elas é a forma como são definidas as associações. Na DML as associações são descritas fora das entidades, onde é pormenorizada a multiplicidade de cada uma das entidades da relação.

A modelação das entidades de domínio começa na DSL, que através do compilador da *framework OMINS* cria a DML. O compilador da *framework* Fenix utiliza a DML para gerar as classes de domínio e as respetivas classes *base*. Estas são utilizadas na camada de persistência, através da *framework* Fenix.

Como referido anteriormente, o processo começa com a modelação do domínio na DSL, como é apresentado no exemplo da Listagem 2.1. Neste exemplo a entidade *ExternalFenixULCompetenceCourse* estende a entidade *CompetenceCourse*. São ainda definidos os atributos de domínio e as associações entre entidades. Esta entidade tem uma associação de muitos para um com a entidade *ExternalInstituitionCommunication*.

```
(...)
entity academicDomainExtensions.ExternalFenixULCompetenceCourse
                                    extends CompetenceCourse channels (WebJava) {
    String externalCourseCode;
    DateTime lastUpdateCourseInformation;
    manyToOne ExternalInstituitionCommunication externalInstituitionCommunication;
}
(...)
```

Listagem 2.1: DSL da classe ExternalFenixULExecutionCourse.

A *framework OMNIS* utiliza a DSL para criar a DML, como exemplificado na Listagem 2.2. Nestes exemplos pode-se observar que enquanto na DSL as associações são definidas dentro das entidades, na DML são definidas fora.

```
(...)
external class .org.fenixedu.academic.domain.CompetenceCourse;
external class .org.fenixedu.academic.domain.ExecutionCourse;
(...)
class academicDomainExtensions.ExternalFenixULCompetenceCourse
                    extends .org.fenixedu.academic.domain.CompetenceCourse  {
    String externalCourseCode ;
    DateTime lastUpdateCourseInformation ;
}
(...)
relation
  ExternalInstituitionCommunicationcompetenceCourseExternalFenixULCompetenceCourse {
    academicDomainExtensions.ExternalFenixULCompetenceCourse playsRole competenceCourse{
        multiplicity *;
    }
    ExternalInstituitionCommunication playsRole externalInstituitionCommunication;
}
(...)
```

Listagem 2.2: DML da classe ExternalFenixULExecutionCourse.

Ainda na compilação do projeto, a *framework* Fenix gera as classes de domínio e as respetivas classes *base*. A classe *base* da *ExternalFenixULCompetenceCourse* é apresentada no exemplo da Listagem 2.3. Esta é gerada com os construtores e os *getters* e *setters* dos atributos e das associações da entidade. A *base* não deve ser alterada, porque esta classe é gerada sempre que o projeto é compilado, de forma a integrar associações que possam ser adicionadas pela DML de outros módulos.

```
public abstract class ExternalFenixULCompetenceCourse_Base extends CompetenceCourse {
    (...)
    // Constructors
    protected ExternalFenixULCompetenceCourse_Base() {
        super();
    }

    // Getters and Setters
    public java.lang.String getExternalCourseCode() {
        (...)
    }

    public void setExternalCourseCode(java.lang.String externalCourseCode) {
         (...)
    }
```

```
16
17      public org.joda.time.DateTime getLastUpdateCourseInformation() {
18          (...)
19      }
20
21      public void setLastUpdateCourseInformation(DateTime lastUpdateCourseInformation) {
22          (...)
23      }
24
25      // Relations Methods
26      public ExternalInstituitionCommunication getExternalInstituitionCommunication() {
27          (...)
28      }
29
30      public void setExternalInstituitionCommunication
31              (ExternalInstituitionCommunication externalInstituitionCommunication) {
32          (...)
33      }
34      (...)
35  }
```

Listagem 2.3: Classe base gerada pela DML para a classe ExternalFenixULCompetenceCourse

2.2.2 Linguagem de modelação de apresentação

A modelação da apresentação pode realizada com a PSL da *framework OMNIS*. O compilador da *framework* gera as classes dos ecrãs e as respetivas classes base.

O processo começa com a modelação dos ecrãs e do fluxo (feito com a PSL), exemplificado na Listagem 2.4. Os ecrãs são descritos dentro de um fluxo. A descrição inclui a indicação do tipo de ecrã, o nome, o tipo de objeto e os campos que o ecrã apresenta. Dentro de um ecrã são criados eventos e ações, que vão estar presentes no ecrã com a identificação do evento e o ecrã para onde o fluxo avança.

```
1  (...)
2  flow manageExternalFenixULPeriodicity channel(WebJava) {
3      searchScreen searchExternalFenixULPeriodicity [ExternalFenixULPeriodicity]
4                                          (fields name time){
5          dialog createEvent -> createExternalFenixULPeriodicity
6          dialog viewAction -> updateExternalFenixULPeriodicity
7          deleteAction
8      }
9
10     createScreen createExternalFenixULPeriodicity [ExternalFenixULPeriodicity]
11                                         (fields name timeUnit timeValue){
12         createEvent
13         cancelEvent
14     }
15
16     updateScreen updateExternalFenixULPeriodicity [ExternalFenixULPeriodicity]
17                                         (fields name timeUnit timeValue) {
18         updateEvent
19         cancelEvent
20     }
21 }
22 (...)
```

Listagem 2.4: Excerto da PSL do projeto.

A *framework OMNIS* utiliza a PSL para gerar as classes dos ecrãs e as respetivas *base*. Na Listagem 2.5 é apresentado um exemplo de uma classe *base* gerada. A classe base contém os campos e os fluxos do ecrã, bem como os métodos utilizados para a criação gráfica dos ecrãs. Para personalizar algumas partes do ecrã ou procedimento, é necessário alterar os respetivos métodos. Posteriormente, estas classes são utilizadas pela *framework Vaadin* [11] para gerar a interface.

```
1  public abstract class CreateExternalFenixULPeriodicity_Base
2                                extends  CreateScreen<ExternalFenixULPeriodicity>  {
3      (...)
4      static {
5          // fields
6          fields.add(ExternalFenixULPeriodicity.META().NAME());
7          fields.add(ExternalFenixULPeriodicity.META().TIME_UNIT());
8          fields.add(ExternalFenixULPeriodicity.META().TIME_VALUE());
9          // navigation
10         navigations.add(new NavigationInformation("CreateEvent",
11                                                   new Class[]{}, false, true));
12         navigations.add(new NavigationInformation("CancelEvent",
13                                                   new Class[]{}, false, true));
14     }
15     (...)
16     @Override
17     public String getFlow() {
18         return "manageExternalFenixULPeriodicity";
19     }
20     (...)
21     protected void processCreateEvent(ClickEvent event) {}
22     (...)
23     protected void processCancelEvent(ClickEvent event) {}
24     (...)
25     @Override
26     protected Schema getSchema() {
27         (...)
28     }
29 }
```

Listagem 2.5: Classe base gerada pela PSL para o ecrã CreateExternalFenixULPeriodicity

2.3 Arquitetura

A arquitetura do sistema Fenix segue o padrão arquitetural *Model-View-Controller* (MVC). Este padrão divide a arquitetura em três camadas logicamente separadas, a camada de persistência que é responsável pela correspondência entre os objetos e as tabelas da base de dados, a camada de aplicação que é responsável por gerir os objetos, os seus procedimentos, entre outros, e a camada de apresentação que é responsável pela interface gráfica do sistema. A utilização deste padrão permite modificar cada uma das camadas de forma independente tendo em atenção que a interface de comunicação entre elas deve permanecer a mesma.

O Fenix é um sistema configurável, por esta razão é um sistema construído de forma modular, onde um conjunto de módulos forma o sistema base do Fenix. A esta base podem ser adicionados módulos, à medida do que são pretendidos para cada instância.

Cada módulo segue o padrão arquitetural do sistema.

De seguida são detalhadas cada uma das camadas [1]:

Camada de persistência

A camada de persistência é abstraída pela *framework* Fenix e assegura duas funcionalidades principais: a persistência dos objetos e a gestão da concorrência de escritas e leituras.

No que diz respeito à persistência dos objetos, a abstração é feita em dois momentos. Durante a compilação do projeto e a inicialização do servidor, a *framework* cria as classes Java das entidades definidas na DML e cria as tabelas das entidades e respetivas associações na base de dados. O segundo momento é durante a execução do Fenix, onde as operações de *Create, Read, Update and Delete* (CRUD) realizadas sobre os objetos de domínio são propagadas para a base de dados de forma transparente para o programador.

No que diz respeito à gestão de concorrência, esta é feita através da JVSTM que garante a coerência nas leituras e nas escritas dos objetos. A JVSTM utiliza caixas de versões que guardam o estado mutável de um programa, no caso do Fenix guardam os atributos mutáveis de cada uma das entidades descritas na DML.

As caixas de versões guardam o valor e a versão das alterações realizadas nos atributos. Este valor não é modificável, quando existe uma nova escrita, é criada uma nova caixa de versão. Isto permite que diferentes *threads* possam aceder a caixas com versões diferentes.

Durante uma transação é utilizada apenas uma caixa de versão, desta forma a modificação de um valor não é visível para as outras *threads*. Durante a execução do *commit* são tratados os possíveis conflitos de acesso ao mesmo objeto. Quando a validação realizada no *commit* falha a transação é reiniciada.

Camada de aplicação

A camada de aplicação é responsável pelas regras de negócio. Estes vão de comportamentos específicos das entidades até ao tratamento dos pedidos feitos ao sistema. É esta camada que faz a ligação entre a camada de apresentação e a camada de persistência. Recebe os pedidos da camada de apresentação, faz o tratamento dos mesmos, onde pode fazer pedidos de leitura ou escrita à camada de persistência, depois do tratamento dos pedidos responde à camada de apresentação com os dados para a interface.

Camada de Apresentação

A camada de apresentação, como o próprio nome indica, é a camada responsável pela interface com o utilizador. Parte desta camada é abstraída pela PSL da *framework OMNIS*.

2.4 Sumário

Neste capítulo foram descritos os conceitos do Fenix, as tecnologias utilizadas, as linguagens das *frameworks* Fenix e *OMNIS* a sua arquitetura. Este conceitos servem de base para o trabalho documentado nos capítulos seguintes.

Capítulo 3

Levantamento de requisitos

Neste capítulo são apresentadas as entidades envolvidas no levantamento de requisitos, os cursos que foram considerados casos de estudo, os atores que irão utilizar o sistema e os casos de uso do estudo que descrevem a forma como atualmente são geridos os cursos partilhados.

3.1 Entidades envolvidas

O levantamento de requisitos foi realizado com a colaboração das seguintes entidades:

- **DI SC** - O Departamento de Informática dos Serviços Centrais (DI SC) é responsável pelo desenvolvimento e manutenção das funcionalidades que garantem a gestão dos cursos partilhados, bem como pelo suporte de segunda linha do sistema Fenix.
- **Faculdade de Letras da ULisboa** - A Faculdade de Letras da ULisboa é responsável por gerir a Licenciatura em Estudos Gerais, um curso partilhado em associação com oito escolas. Por esta razão é um dos parceiros no levantamento de requisitos.
- **Instituto de Geografia e Ordenamento do Território** - O Instituto de Geografia e Ordenamento do Território é responsável por gerir cursos partilhados de várias tipologias, isto é, em associação, co-tutela e de gestão rotativa. Este facto justifica também a sua presença no levantamento de requisitos.

3.2 Casos de estudo

Os cursos apresentados de seguida foram utilizados como casos de estudo, sendo cursos das diferentes tipologias de cursos partilhados.

- **Licenciatura em Estudos Gerais (Associação)** - A licenciatura em Estudos Gerais é um curso em associação com o tronco comum na Faculdade de Letras da ULisboa e com possibilidade de realização de *majors/minors* nas seguintes sete escolas da

ULisboa: Faculdade de Belas-Artes, Faculdade de Ciências, Faculdade de Direito, Faculdade de Motricidade Humana, Faculdade de Psicologia, Instituto Superior de Ciências Sociais e Políticas e Instituto Superior de Economia e Gestão.

- **Mestrado em Ensino de Geografia (Co-tutela)** - O Mestrado em Ensino de Geografia é gerido em co-tutela pelas seguintes escolas da ULisboa: Instituto de Educação e Instituto de Geografia e Ordenamento do Território.

- **Mestrado em Ordenamento do Território e Urbanismo (Gestão rotativa)** - O mestrado em Ordenamento do Território e Urbanismo é gerido de forma rotativa pelas seguintes escolas da ULisboa: Instituto Superior Técnico, Instituto de Geografia e Ordenamento do Território e Faculdade de Arquitetura.

3.3 Atores

Os atores envolvidos nos casos de usos do levantamento de requisitos são:

- **Aluno** - Aluno de um curso partilhado.

- **Serviços Académicos (SA)-Origem** - Serviços Académicos da escola de origem, responsável pela gestão académica na escola de origem do curso.

- **SA-Destino** - Serviços Académicos da escola de destino, responsável pela gestão académica na escola de destino do curso.

- **Serviços Financeiros** - Serviços Financeiros das escolas, responsáveis pela área financeira das escolas de origem ou destino.

- **Professor** - Professor na escola de destino, responsável pela gestão da informação das suas UCs e o lançamento das pautas das mesmas. Os professores da escola de origem não têm impacto no âmbito do projeto.

3.4 Levantamento de requisitos: Casos de uso

O levantamento de requisitos originou os casos de uso da forma como atualmente são geridos os cursos partilhados na ULisboa. Na Tabela 3.1 são listados os casos de uso que serão apresentados posteriormente, juntamente com as respetivas alternativas e o nível de detalhe da descrição.

Tabela 3.1: Casos de uso do levantamento de requisitos

Caso de Uso do Estudo (CUE) e respetivas Alternativas (A)		Descrição
LR:CU1	Criar UCs na escola de destino	Detalhada
LR:CU1-A1	As UCs são associadas ao curso livre	Detalhada
LR:CU2	Criar curso na escola de origem incluindo UCs externas e horários	Detalhada
LR:CU3	Matrícula do aluno	Detalhada
LR:CU4	Inscrição do aluno	Detalhada
LR:CU4-A1	Reinscrição de um aluno de um curso em gestão rotativa que interrompeu os estudos	Detalhada
LR:CU4-A2	Caso a escola não informe a escola de destino dos alunos que se inscrevem nas suas UCs	Detalhada
LR:CU4-A3	Não existem vagas na escola de destino para o turno escolhido	Detalhada
LR:CU5	Alteração das inscrições	Detalhada
LR:CU5-A1	Caso a escola não tenha informado a escola de destino dos alunos que se inscreveram nas suas UCs	Detalhada
LR:CU5-A2	Caso o aluno apenas pretenda apenas alterar os turnos/turmas	Detalhada
LR:CU6	Transferência de escola de um aluno (sem alteração de curso)	Detalhada
LR:CU7	Inscrição a avaliações	Detalhada
LR:CU7-A1	Caso a escola gira todas as UCs no seu Fenix	Detalhada
LR:CU8	Lançar a pauta de uma UC com alunos de cursos partilhados	Detalhada
LR:CU8-A1	O professor lança a pauta nas duas escolas	Detalhada
LR:CU9	Inquéritos às UCs	Breve
LR:CU10	Pedido de documentos académicos	Breve
LR:CU11	Visualizar o horário do aluno	Detalhada
LR:CU11-A1	No caso de alunos com UCs em escolas que não usem o Fenix	Detalhada
LR:CU12	Visualizar a informação da página pública do curso e das UCs	Detalhada
LR:CU13	Visualizar a conta corrente do aluno	Detalhada
LR:CU14	Candidatura de mobilidade *outgoing* com acordo da escola de destino	Detalhada
LR:CU15	Criar relatório para RAIDES	Detalhada
LR:CU15-A1	No caso dos alunos serem divididos pelas diferentes escolas do curso	Detalhada
LR:CU16	Candidatura a bolsas SAS	Breve
LR:CU17	Acerto de contas	Breve

LR:CU1 - Criar UCs na escola de destino

Nome: Criar UCs na escola de destino

Ator Principal: SA-Destino

Período: Preparação do ano letivo.

Pré-Condições:
- O curso foi criado em Diário da República.
- As UCs são oferta no ano letivo corrente.

Pós-Condições:
- As UCs são criadas na escola de destino.
- As UCs ficam em execução para o ano letivo corrente.

Cenário Principal de Sucesso:

1- Os SA-Destino utilizam a informação sobre as UCs lecionadas na sua escola, que estão em despacho do curso.

2- Os SA-Destino criam as UCs na instância Fenix com os mesmos parâmetros.

3- Os SA-Destino criam o curso.

4- Os SA-Destino criam um plano curricular para o curso criado no ponto anterior apenas com as UCs lecionadas nesta escola.

5- Os SA-Destino criam uma execução do curso para o ano letivo corrente.

6- Os SA-Destino criam uma execução das UCs para o ano letivo corrente.

7- Nos anos seguintes os SA-Destino abrem e fecham as execuções das UCs, caso sejam ou não lecionadas.

Cenários Alternativos:

LR:CU1-A1: 3a - As UCs são associadas ao curso livre.

1 - Os SA-Destino adicionam estas UCs ao plano curricular do curso livre, utilizado para realização de UCs isoladas.

2 - O fluxo continua no ponto 6.

Problemas inerentes:

- **Correspondência das UCs** - Neste momento, não existe no Sistema Fenix uma correspondência entre as UCs reais da escola de destino com a cópia das UCs na escola de origem. Esta correspondência é feita através de ficheiros excel.
- **Duplicação de dados** - No caso do curso ser criado na escola de destino, o curso existe em duplicado nas duas instâncias Fenix.

LR:CU2 - Criar curso na escola de origem incluindo UCs externas e horários

Nome: Criar curso na escola de origem incluindo UCs externas e horários

Ator Principal: SA-Origem

Atores Secundários: SA-Destino

Período: Preparação do ano letivo

Pré-Condições:
- O curso foi criado em Diário da República.
- As UCs são oferta no ano letivo corrente.

Pós-Condições:
- O Curso, o plano e as UCs são criadas.
- O Curso e as UCs ficam em execução.
- Os Horários das UCs externas ficam carregados no Fenix.

Cenário Principal de Sucesso:

1- Os SA-Origem utilizam a informação sobre as UCs que está em despacho do curso.

2- Os SA-Origem criam as UCs na instância Fenix com os mesmos parâmetros.

3- Os SA-Origem criam o curso.

4- Os SA-Origem criam um plano curricular para o curso criado no ponto anterior.

5- Os SA-Origem criam uma execução do curso para o ano letivo corrente.

6- Os SA-Origem pedem a Distribuição de Serviço Docente aos SA-Destino com a oferta formativa, horários, entre outras informações.

7- Os SA-Origem inserem a informação pedida no Fenix de origem.

8- Os SA-Origem criam uma execução das UCs para o ano letivo corrente.

Problemas inerentes:

- **Inserção de erros humanos** - Ao criar a cópia da UC no Fenix na escola de origem, os SA podem inserir erros no código, nome, European Credit Transfer System (ECTS), entre outros.
- **Correspondência das UCs** - Neste momento, não existe no Sistema Fenix uma correspondência entre as UCs reais da escola de destino com a cópia das

UCs na escola de origem. Esta correspondência é feita através de ficheiros excel.

- **Modificação das UCs na escola de destino** - A reestruturação de cursos onde as UCs estão inseridas na escola de destino pode originar a alteração do nome/código das UCs, bem como dos ECTS. No entanto, no âmbito dos cursos partilhados, devem ser mantidos iguais ao que se encontra no despacho publicado em Diário da República.
- **Duplicação de dados** - A criação de cópias de UCs no Fenix de origem implica que as informações básicas das UCs sejam duplicadas nas duas instâncias Fenix, bem como horários.
- **Comunicação entre SA** - Os SA têm de comunicar para pedir a Distribuição de Serviço Docente.

LR:CU3 - Matrícula do aluno
Nome: Matrícula do aluno
Ator Principal: Aluno
Período: Durante o período de matrículas
Pré-Condições:
- Os SA-Origem importaram o ficheiro de colocados no Fenix.
- O aluno está colocado no curso.
- O aluno não está matriculado no curso.

Pós-Condições:
- O aluno fica matriculado no curso.

Cenário Principal de Sucesso:

1- O aluno inicia o processo de matrícula.
2- O aluno preenche o inquérito de matrícula.
3- O aluno prossegue para a inscrição no **LR:CU4**.

LR:CU4 - Inscrição do aluno
Nome: Inscrição do aluno
Ator Principal: Aluno
Atores Secundários: SA-Origem, SA-Destino
Período: Durante o período de matrículas
Pré-Condições:
- O aluno está matriculado.

Pós-Condições:
- O aluno fica inscrito no ano corrente e em todas as UCs.

Cenário Principal de Sucesso:

1- O aluno inicia o processo de inscrição.
2- O sistema fornece ao aluno todas as UCs do seu curso.
3- O aluno escolhe as UCs que pretende realizar nesse ano curricular.
4- O aluno escolhe o turno e o horário de cada uma das UCs.
5- O aluno finaliza a inscrição.
6- Os SA-Origem confirmam a inscrição do aluno.

7- Os SA-Origem da escola onde o aluno está matriculado enviam a informação do aluno, a sua escolha de turnos e horários para os SA-Destino das escolas onde estas UCs são lecionadas.

8- Os SA-Destino recebem a informação do aluno e criam o aluno na sua instância Fenix

9- Os SA-Destino inscrevem o aluno nas UCs, nos turnos e nos horários que este escolheu.

Cenários Alternativos:

LR:CU4-A1: 0a - Reingresso de um aluno de um curso em gestão rotativa que interrompeu os estudos

1 - O aluno deve realizar a matrícula (Caso de uso **LR:CU3**) na escola onde o curso está a ser gerido nesse ano.

2 - O aluno continua o processo de inscrição no ponto 1 na nova escola.

LR:CU4-A2: 6a - Caso a escola não informe a escola de destino dos alunos que se inscrevem nas suas UCs

1 - O processo termina neste ponto.

LR:CU4-A3: 9a - Não existem vagas na escola de destino para os turnos que o aluno escolheu

1 - Os SA-Destino avisam os SA-Origem que o turno escolhido pelo aluno não tem vagas e fornecem alternativas.

2 - Os SA-Origem avisam o aluno que o turno não tem vagas e fornecem alternativas.

3 - O aluno escolhe o novo turno

4 - Os SA-Origem enviam a novas informações para os SA-Destino.

5 - O fluxo continua no ponto 9.

Problemas inerentes:

- **Comunicação entre SA** - Os SA têm de comunicar para passar a informação do aluno da escola de origem para a escola de destino.
- **Inserção de erros humanos** - A passagem da informação do aluno entre as escolas pode levar à inserção de erros.
- **Demora do processo** - Entre o fim da inscrição por parte do aluno na escola de origem e a inscrição do aluno na escola de destino existe uma grande demora do processo, podendo demorar semanas. Um aluno que se inscreva em setembro na sua escola de origem, pode só ser inscrito na UC externa depois do início do período letivo.
- **Diferenças do calendário académico** - Os períodos de inscrição diferentes entre as diversas escolas pode levar a problemas nas inscrições. Por exemplo, num cenário em que uma escola realiza inscrições anuais e uma escola de destino de uma UC realiza as inscrições semestrais, os alunos que se pretendam

inscrever-se anualmente a todas as UCs, não o poderão fazer. Neste caso terão de se inscrever pessoalmente junto dos seus SA no segundo semestre. Adicionalmente, se se verificar incompatibilidade entre os horários, os alunos serão obrigados a alterar a sua inscrição

LR:CU5 - Alteração das inscrições
Nome: Alteração das inscrições
Ator Principal: Aluno
Atores Secundários: SA-Origem, SA-Destino
Período: Durante as épocas de alteração das inscrições
Pré-Condições:
- O Aluno está matriculado.
- O Aluno está inscrito na UC

Pós-Condições:
- O aluno fica com a inscrição nas UCs alterada.

Cenário Principal de Sucesso:

1- O aluno pede aos SA-Origem para fazer alteração da inscrição.
2- O aluno escolhe as UCs que pretende alterar.
3- Os SA-Origem informam os SA da escola responsável pela UC que o aluno cancelou a inscrição.
4- Os SA-Destino cancelam a inscrição do aluno nas UCs indicadas.
5- Os SA-Origem enviam as informações e as UCs que o aluno escolheu para os SA-Destino.
6- Os SA-Destino matriculam o aluno se ele ainda não estiver matriculado e inscreve o aluno na UC.

Cenários Alternativos:

LR:CU5-A1: 3a - Caso a escola não tenha informado a escola de destino dos alunos que se inscreveram nas suas UCs

1 - Os SA-Origem cancelam a inscrição do aluno nas UCs escolhidas pelo aluno.
2 - Os SA-Origem inscrevem os alunos nas novas UCs.

LR:CU5-A2: 2a - Caso o aluno apenas pretenda alterar os turnos/turmas.

1 - O aluno escolhe os turnos que pretende alterar.
2 - Os SA-Origem informam os SA da escola responsável pela UC que o aluno alterou o turno/turma.

Problemas inerentes:

- **Comunicação entre SA** - Os SA têm de comunicar para passar as informações da anulação da inscrição das UCs, bem como da nova inscrição.
- **Inserção de erros humanos** - Na passagem da informação do aluno entre as escolas pode levar a inserção de erros.
- **Demora do processo** - Entre a reinscrição de um aluno na escola de origem e a nova inscrição na escola de destino pode levar bastante tempo.

- **Diferenças de calendário académico** - Os períodos de inscrição diferentes entre as escolas pode levar a problemas nas inscrições. Por exemplo, uma escola que realiza inscrições anuais e a escola de destino de uma UC que realiza as inscrições semestrais, neste caso um aluno que se pretenda inscrever anualmente, como faz com as outras UCs, não o poderá.

LR:CU6 - Transferência de escola de um aluno (sem alteração de curso)
Nome: Transferência de escola de um aluno (sem alteração de curso)
Ator Principal: SA-Origem
Atores Secundários: SA-Destino
Período: Preparação do ano letivo.
Pré-Condições:
- O aluno transitou para o ano seguinte.

Pós-Condições:
- O aluno fica inscrito na escola de destino
- A escola de origem passa a ser a escola de destino.
- A escola de destino passa a ser a escola de origem.

Cenário Principal de Sucesso:
1- Os SA-Origem digitaliza o processo do aluno.
2- Os SA-Origem enviam o processo do aluno para os SA-Destino.
3- Os SA-Destino matriculam o aluno na sua escola.
4- Os SA-Destino lançam as notas do aluno no seu Fenix de forma avulsa.

Problemas inerentes:
- **Comunicação entre SA** - Os SA têm de comunicar para passar as informações do aluno.
- **Inserção de erros humanos** - A passagem da informação do aluno entre as escolas pode levar a inserção de erros.
- **Lançamento de notas avulsas** - Os SA lançam as notas do aluno de forma avulsa.
- **Duplicação de dados** - As notas dos alunos ficam duplicadas nas escolas.

LR:CU7 - Inscrição a avaliações
Nome: Inscrição a avaliações
Ator Principal: Aluno
Período: Durante as épocas de inscrição a avaliações
Pré-Condições:
- O aluno está inscrito à UC.

Pós-Condições:
- O aluno fica inscrito à avaliação da UC.

Cenário Principal de Sucesso:
1- O aluno faz a inscrição nos momentos de avaliação das UCs no Fenix da escola de origem.
2- O aluno faz a inscrição nos momentos de avaliação das UCs no Fenix das escolas de destinos.

Cenários Alternativos:

LR:CU7-A1: 2a - Caso a escola gira todas as UCs no seu Fenix

1 - O aluno pode-se inscrever em todas as avaliações na sua instância Fenix.

Problemas inerentes:

- **Acesso a diferentes instâncias Fenix** - Para o aluno se inscrever em todas as avaliações das UCs que está a realizar deve aceder às instâncias das escolas onde está a realizar UCs.
- **Divisão da informação** - Na escola de origem, o aluno não tem informação das épocas de avaliação que realizou.
- **Diferente nomenclatura** - O nome das épocas de avaliação podem não existir na escola de origem, ficando assim as avaliações numa época diferente à original.

LR:CU8 - Lançar a pauta de uma UC com alunos de cursos partilhados

Nome: Lançar a pauta de uma UC com alunos de cursos partilhados

Ator Principal: Professor

Atores Secundários: SA-Origem, SA-Destino

Período: Durante as épocas de avaliação

Pré-Condições: • O aluno está inscrito.

Pós-Condições: • O aluno fica com o seu currículo na escola de origem atualizado.

Cenário Principal de Sucesso:

1- O professor lança a pauta da sua UC, na instância Fenix da sua escola.

2- Os SA-Destino verificam a pauta.

3- Os SA-Destino enviam as notas dos alunos de cursos partilhados para as suas escolas.

4- Os SA-Origem recebem a nota do aluno.

5- Os SA-Origem lançam a nota no currículo do aluno como nota avulsa.

Cenários Alternativos:

LR:CU8-A1: 2a - O professor lança a pauta nas duas escolas

1 - O professor lança a pauta na escola de origem para os alunos matriculados na escola de origem.

2 - Os SA-Origem verificam a pauta.

Problemas inerentes:

- **Comunicação entre SA** - Os SA têm de comunicar para passar a pauta da escola de destino para a escola de origem.
- **Inserção de erros Humanos** - Os SA podem introduzir erros como uma alteração da nota ou a troca da nota com outro aluno, tanto no envio da pauta da escola de destino para a escola de origem, como no lançamento das notas avulsas.
- **Demora do Processo** - Entre a pauta ser lançada e a nota estar disponível no currículo do aluno na escola origem pode demorar até três meses.

- **Transparência da nota** - Na escola de origem o aluno não consegue visualizar a pauta na qual a sua nota foi lançada.
- **Duplicação de dados** - A notas são lançadas nas duas escolas.
- **Lançamento de notas avulsas** - A notas são lançadas como notas avulsas na escola de origem.
- **Diferente nomenclatura** - O nome das épocas de avaliação podem não existir na escola de origem, ficando assim as avaliações lançadas numa época diferente à original.
- **Acesso a diferentes instâncias Fenix** - No caso da alternativa **LR:CU7-A1**, o professor tem de aceder a duas instâncias Fenix para poder lançar as notas.

LR:CU9 - Inquéritos às UCs

Nome: Inquéritos às UCs

Ator Principal: Aluno

Período: Momento de Inquéritos

Descrição:

No período de resposta a Inquéritos estes são lançados no Fenix das escolas onde as UCs são lecionadas.No entanto os alunos de cursos partilhados não costumam responder a esses inquéritos, porque quando estes são lançados os alunos já não se encontram nessa escola a realizar UCs.

No caso **LR:CU1**, se as UCs tiverem sido criadas e inseridas num curso livre, o aluno não terá inquéritos disponíveis para essas UCs, porque os alunos de cursos livres não são contabilizados para questões de inquéritos.

Problemas inerentes:

- **Acesso a diferentes instâncias Fenix** - Para o aluno realizar os inquéritos, o aluno deve aceder à instância Fenix onde a UC é lecionada.

LR:CU10 - Pedido de documentos académicos

Nome: Pedido de documentos académicos

Ator Principal: Aluno

Atores Secundários: SA-Origem

Período: Durante todo o ano

Descrição:

O aluno pede um documento académico na sua escola de origem. Caso o documento apenas diga respeito a informações dessa escola, os SA criam o documento e entregam-no ao aluno.

Caso nenhuma da informação diga respeito a essa escola, o aluno precisa de ir fazer esse pedido à escola onde existe essa informação.

Caso a informação esteja dividida entre as escolas, como pode ser o caso do currículo do aluno ou dos programas curriculares, os SA-Origem pedem essa informação às outras escolas, agregam a informação, criam o documento e entregam-no ao aluno.

Problemas inerentes:

- **Comunicação entre SA** - Os SA têm de comunicar, caso as informações estejam divididas.
- **Inserção de erros Humanos** - Na agregação dos dados, as informações podem ser inseridas com erros.
- **Demora do Processo** - Caso a informação se encontre dividida, o processo pode ser bastante mais longo que o normal.

LR:CU11 - Visualizar o horário do aluno
Nome: Visualizar o horário do aluno
Ator Principal: Aluno
Período: Ao longo do ano letivo
Pré-Condições: • Os horários devem ter sido lançados.
Cenário Principal de Sucesso:

1- O aluno visualiza o seu horário na instância Fenix da sua escola de origem.

Cenários Alternativos:
LR:CU11-A1: 3a - No caso de alunos com UCs em escolas que não usem o Fenix

1 - O aluno visualiza o horário das UCs lecionadas nestas escolas no respetivo SIGA.

Problemas inerentes:

- **Horários fora do Fenix** - Atualmente o aluno não consegue visualizar as UCs das escolas que tem o horário fora do Fenix
- **Duplicação de dados** - Os horários visualizados pelo aluno encontram-se duplicados em várias instâncias Fenix, como referido no Caso de uso **LR:CU2**

LR:CU12 - Visualizar a informação da página pública do curso e das UCs
Nome: Visualizar a informação da página pública do curso e das UCs
Ator Principal: Aluno
Período: Durante todo o ano
Cenário Principal de Sucesso:

1- O aluno visualiza a informação do curso na escola responsável pelo mesmo.
2- O aluno visualiza a informação das UCs no Fenix da escola onde são lecionadas.

Problemas inerentes:

- **Acesso a diferentes instâncias Fenix** - Para o aluno visualizar toda a informação relativa a um curso, o aluno precisa de procurar a informação nas diversas instâncias Fenix.

LR:CU13 - Visualizar a conta corrente do aluno
Nome: Visualizar a conta corrente do aluno
Ator Principal: Aluno

Período: Ao longo do ano letivo

Nota: Tipicamente o aluno apenas tem a necessidade de aceder a diferentes Fenix para ver emolumentos específicos ou propinas antigas, se o aluno tiver sido transferido.

Cenário Principal de Sucesso:

1- O aluno visualiza a sua conta corrente no Fenix da sua escola de origem.

2- O aluno visualiza a sua conta corrente no Fenix da escola de destino.

Problemas inerentes:

- **Acesso a diferentes instâncias Fenix** - O aluno apenas consegue ver a conta corrente referente à escola da instância onde se encontra.

LR:CU14 - Candidatura de mobilidade *outgoing* com acordo da escola de destino

Nome: Candidatura de mobilidade *outgoing* com acordo da escola de destino

Ator Principal: Aluno

Período: Durante o período de candidaturas de mobilidade *outgoing*

Pré-Condições:
- O período de candidaturas está aberto.

Pós-Condições:
- O aluno fica com a candidatura criada.

Cenário Principal de Sucesso:

1- O aluno informa a escola de origem que pretende realizar mobilidade *outgoing*.

2- O aluno cria um processo de candidatura *outgoing* no Fenix da escola responsável pelo acordo.

3- O coordenador Erasmus/DREI da escola responsável pelo acordo avaliam o processo.

4- Os coordenadores Erasmus/DREI das escolas responsáveis pelas UCs avaliam as correspondências.

5- O aluno é seriado.

Problemas inerentes:

- **Demora do processo** - No caso do aluno escolher UCs de escolas que não são da escola responsável pelo acordo, como as correspondências das UCs são avaliados pelos coordenadores Erasmus/DREI da escola responsável pelas UCs, o processo é enviado para essas escolas, o que leva à demora do processo.
- **Inserção de erros humanos** - Na troca de documentos/informação entre as diversas escolas podem ser inseridos erros.
- **Comunicação entre SA** - Na troca de documentos/informação é realizada por comunicação fora do Fenix.
- **Diferenças do calendário académico** - Os períodos de candidatura e de seriação são diferentes entre escolas, o que pode levar a problemas do aluno na fase de confirmação da escolha do acordo. Por exemplo se a escola B acabar a seriação antes da escola A e o aluno preferir um acordo da escola A, o aluno terá de confirmar ou rejeitar o acordo antes de saber se foi seriado para fazer mobilidade pelo acordo da escola A.

LR:CU15 - Criar relatório para RAIDES

Nome: Criar relatório para RAIDES

Ator Principal: SA-Origem

Período: Em dois momentos no ano letivo, o primeiro a 31 de dezembro e o segundo a 31 de março

Pré-Condições:
- O curso faz parte do âmbito do relatório.
- Os alunos inscritos/diplomados fazem parte do âmbito do relatório.

Pós-Condições:
- Os documentos para o relatório são gerados.

Cenário Principal de Sucesso:

1- Os SA atualizam a configuração dos RAIDES.
2- Os SA criam um pedido de criação de relatório.
3- Os SA fazem download dos relatórios.
4- Os SA verificam os dados.
5- Os SA entregam o relatório na Plataforma de Recolha de Informação do Ensino Superior (PRIES).

Cenários Alternativos:

LR:CU15-A1: 5a - No caso dos alunos serem divididos pelas diferentes escolas do curso

1 - O fluxo deve ser realizado por cada uma das escolas do curso

Problemas inerentes:

- **Alunos duplicados** - Os alunos podem ser reportados em diferentes escolas no mesmo curso.
- **Alunos transferidos** - Os alunos transferidos de escola no âmbito do mesmo curso e reportados nos dois anos, implicam pedido de esclarecimentos.
- **Inserção de erros humanos** - No caso da alternativa **LR:CU15-A1** podem ser inseridos erros no preenchimento manual do relatório, bem como na comunicação entre escolas.
- **Comunicação entre SA** - No caso da alternativa **LR:CU15-A1** os SA das escolas comunicam para o envio da informação relativo aos alunos divididos.

LR:CU16 - Candidatura a bolsas SA

Nome: Candidatura a bolsas SA

Ator Principal: Aluno

Atores Secundários: SA-Origem

Período: Antes das matrículas/inscrições

Descrição:

O aluno realiza a candidatura a bolsa através da Direção-Geral de Ensino Superior (DGES), que por sua vez envia a informação/pedido para a escola responsável pelo curso onde o aluno está matriculado. Os SA-Origem preenchem os dados académicos do aluno e reenviam para a DGES.

Problemas inerentes:

- **Dados em falta na origem** - Os dados do aluno podem não existir na escola de origem.

LR:CU17 - Acerto de contas
Nome: Acerto de contas
Ator Principal: Serviços Financeiros
Período: No final do ano letivo.
Descrição:

No final do ano letivo, as escolas realizam o acerto de contas, com movimentos contabilísticos entre os serviços financeiros de cada uma das escolas. Este acerto de contas é realizado fora do Fenix.

No âmbito dos cursos partilhados, a única funcionalidade que poderia ser adicionada ao Fenix seria o cálculo dos valores do acerto, tendo em conta os alunos de cada uma das escolas que esteve a realizar UCs na outra escola, bem como o custo de cada uma das UCs.

O caso dos acertos de contas que são realizados através do número de UCs disponibilizadas no plano curricular, esta funcionalidade não teria grande impacto.

3.5 Análise dos casos de uso do estudo

Na Tabela 3.2 pode ser vista a agregação dos problemas e a sua existência transversal aos casos de uso, evidenciando os casos de uso com maior número de problemas, bem como os problemas que estão associados a mais casos de uso.

Tabela 3.2: Agregação dos problemas encontrados nos casos de uso

Problemas	LR:CU1	LR:CU2	LR:CU3	LR:CU4	LR:CU5	LR:CU6	LR:CU7	LR:CU8	LR:CU9	LR:CU10	LR:CU11	LR:CU12	LR:CU13	LR:CU14	LR:CU15	LR:CU16	LR:CU17
Correspondência da UCs	x	x															
Duplicação de dados	x	x				x					x						
Inserção de erros humanos		x		x	x	x		x		x				x	x		
Modificação das UCs na escola de destino		x															
Comunicação entre SA		x		x	x	x		x		x				x	x		
Demora do processo				x	x			x		x				x			
Diferenças do calendário académico				x	x									x			
Lançamento de notas avulsas						x		x									
Acesso a diferentes instâncias Fenix							x	x	x			x	x				
Divisão da informação							x										
Transparência da nota								x									
Diferente nomenclatura							x	x									
Horários fora do Fenix											x						
Alunos duplicados															x		
Alunos transferidos																	
Dados em falta na origem																x	

Um dos principais problemas que está associado a mais casos de uso é relativo à comunicação entre SA, que exige um maior esforço laboral e está na origem de outros

dois problemas: a inserção de erros humanos e a demora dos processos. O primeiro deve-se ao processamento manual da informação que pode criar erros nos dados. Isto é agravado com a quantidade da informação e com o número de vezes que a informação é registada manualmente. O segundo problema deve-se ao facto de existir uma tentativa de redução de mensagens trocadas, enviando assim o máximo de informação possível, por mensagem. Isto pode levar a que algumas alterações possam demorar várias semanas até estarem disponíveis na instância que recebe os dados.

Outro problema é a necessidade dos utilizadores acederem a várias instâncias para realizar a mesma ação. Isto acontece principalmente com ações referentes à inserção de informação, como inscrições a avaliações, resposta a inquéritos, lançamento de notas, entre outros. Nestas ações, os utilizadores têm de aceder a cada uma das instâncias que são responsáveis pelas UCs.

O último problema transversal a um grande número de casos de uso refere-se à duplicação dos dados nas diferentes instâncias Fenix. Esta duplicação pode levar à falta de coerência dos dados, bem como à extração de dados errados do Fenix, por exemplo quando os SA fazem a extração das UCs da sua escola, extraem também as UCs externas.

Nos restantes problemas existem três que merecem mais atenção, nomeadamente: as correspondências das UCs, as diferenças dos calendários académicos e as diferenças de nomenclaturas. De facto, apesar de estes problemas surgirem em poucos casos de uso, a sua solução apresenta desafios que em alguns casos estão fora do âmbito de um sistema informático, por exemplo as diferenças dos calendários académicos.

3.6 Sumário

Neste capítulo é descrito o levantamento de requisitos, sendo a sua sistematização apresentada através dos casos de uso que representam a forma como é efetuada a gestão dos cursos partilhados *atualmente*.

Capítulo 4

Análise de requisitos

Neste capitulo são apresentados os resultados da análise de requisitos do trabalho realizado através da definição dos casos de uso e dos requisitos não funcionais.

4.1 Introdução

A análise de requisitos é feita tendo em conta os casos de uso do levantamento de requisitos, os quais justificam os seguintes obejetivos transversais:

- **Reduzir a comunicação entre SA** - Desta forma são reduzidos os problemas inerentes a esta comunicação em massa, como a inserção de erros humanos e a demora dos processos. Para tal é necessário que a comunicação passe a ser feita entre instâncias Fenix.
- **Reduzir a duplicação da informação** - A duplicação de informação pode levar a problemas de sincronismo e de coerência da informação nas diversas instâncias.
- **Reduzir o acesso aos diferentes Fenix** - A transparência da informação deve ser mantida de forma a que os utilizadores finais, mais propriamente os alunos de cursos partilhados, possam aceder a apenas uma instância Fenix, tal como os alunos de cursos de uma única escola.

Para além destes objetivos, existem alguns problemas a ter em conta:

- **Modificação das UCs na escola de destino** - A alteração de uma UC na escola de origem deve ser gerida para que a estrutura do curso partilhado se mantenha igual ao que está publicado em Diário da República.
- **Calendários académicos diferentes** - A existência de calendários académicos para as diferentes escolas pode levar a problemas no cumprimento dos prazos dos diversos momentos.

Os casos de uso estão listados na Tabela 4.1. De forma a clarificar as alterações os passos dos cenários principais de sucesso que não sofreram alterações encontram-se a cinzento. Depois de cada caso de uso, é apresentado o respetivo diagrama de sequência.

Para a descrição dos casos de uso é utilizado o pressuposto de que a escola de destino tem uma representação do curso apenas com as UCs que leciona.

Tabela 4.1: Casos de uso da análise de requisitos

Caso de Uso (CU)		Descrição
CU1	Criar UCs na escola de destino	Detalhado
CU2	Criar curso na escola de origem	Detalhado
CU3	Matrícula do aluno	Detalhado
CU4	Inscrição do aluno	Detalhado
CU5	Alteração das inscrições	Detalhado
CU6	Transferência de escola de um aluno (sem alteração de curso)	Detalhado
CU7	Inscrição a avaliações	Detalhado
CU8	Lançar a pauta de uma UC com alunos de cursos partilhados	Detalhado
CU9	Visualização do currículo	Detalhado
CU10	Inquéritos às UCs	Detalhado
CU11	Pedido de documentos académicos	Detalhado
CU12	Visualizar o horário do aluno	Detalhado
CU13	Visualizar o horário de um semestre do curso	Detalhado
CU14	Visualizar a informação da página pública do curso e das UCs	Detalhado
CU15	Visualizar a conta corrente do aluno	Detalhado
CU16	Candidatura a proposta de formação avançada	Detalhado
CU17	Visualização do processo de formação avançada	Detalhado
CU18	Candidatura de mobilidade *outgoing*	Detalhado
CU19	Criar relatório para RAIDES	Detalhado
CU20	Candidatura a bolsas SAS	Detalhado
CU21	Acerto de contas	Breve

4.2 Casos de uso

CU1 - Criar UCs na escola de destino

Nome: Criar UCs na escola de destino

Ator Principal: SA-Destino

Período: Preparação do ano letivo.

Pré-Condições:
- O curso foi criado em Diário da República.
- As UCs são oferta no ano letivo corrente.

Pós-Condições:
- As UCs são criadas na escola de destino.
- As UCs ficam em execução para o ano letivo corrente.

Cenário Principal de Sucesso:

1- Os SA-Destino utilizam a informação sobre as UCs lecionadas na sua escola, que estão em despacho do curso.

2- Os SA-Destino criam as UCs, que são lecionadas nesta escola, na sua instância Fenix.

3- Os SA-Destino criam uma representação do curso.

4- Os SA-Destino associam as UCs à representação do curso.

5- Os SA-Destino criam a execução da representação do curso para o ano letivo corrente.

6- Os SA-Destino criam a execução das UCs para o ano letivo corrente.

7- Nos anos seguintes os SA-Destino abrem e fecham as execuções das UCs, caso sejam ou não lecionadas.

8- Os SA-Destino criam os turnos para as UCs do curso partilhado com as vagas disponíveis para as mesmas.

Diagrama de sequência: Figura 4.1

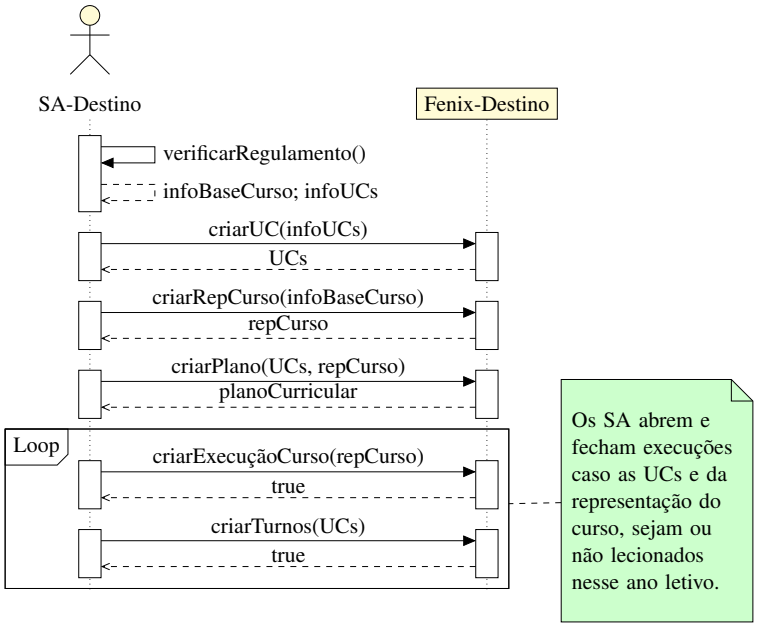

Figura 4.1: Diagrama de sequência do caso de uso de criação das UCs na escola de destino.

CU2 - Criar curso na escola de origem

Nome: Criar curso na escola de origem

Ator Principal: SA-Origem

Atores Secundários: SA-Destino

Período: Preparação do ano letivo

Pré-Condições:
- O curso foi criado em Diário da República.
- As UCs são oferta no ano letivo corrente.

Pós-Condições:
- O Curso, o plano e as UCs são criadas.
- O Curso e as UCs ficam em execução para o ano letivo corrente.

Cenário Principal de Sucesso:

1- Os SA-Destino utilizam a informação sobre as UCs lecionadas na sua escola,

que estão em despacho do curso.

2- Os SA-Origem criam as UCs, que são lecionadas nesta escola, na sua instância Fenix.

3- Os SA-Origem criam o curso.

4- Os SA-Origem criam um plano curricular para o curso.

5- Os SA-Origem associam as UCs externas ao plano.

6- O Fenix-Origem pede às instâncias Fenix-Destino as informações relativas às UCs externas a cada uma das escolas.

7- Os SA-Origem criam uma execução do curso para o ano letivo corrente.

8- Os SA-Origem criam uma execução das UCs para o ano letivo corrente.

9- Nos anos seguintes os SA-Origem abrem e fecham as execuções das UCs, caso sejam ou não lecionadas.

Cenários Alternativos:

CU1-A1: 6a - As UCs externas não estão criadas ou não estão associadas ao curso.

1 - Os SA-Origem pedem aos SA-Destino para criarem as UCs, executando o caso de uso (**CU1**).

2 - O fluxo continua no ponto 5.

Diagrama de sequência: Figura 4.2

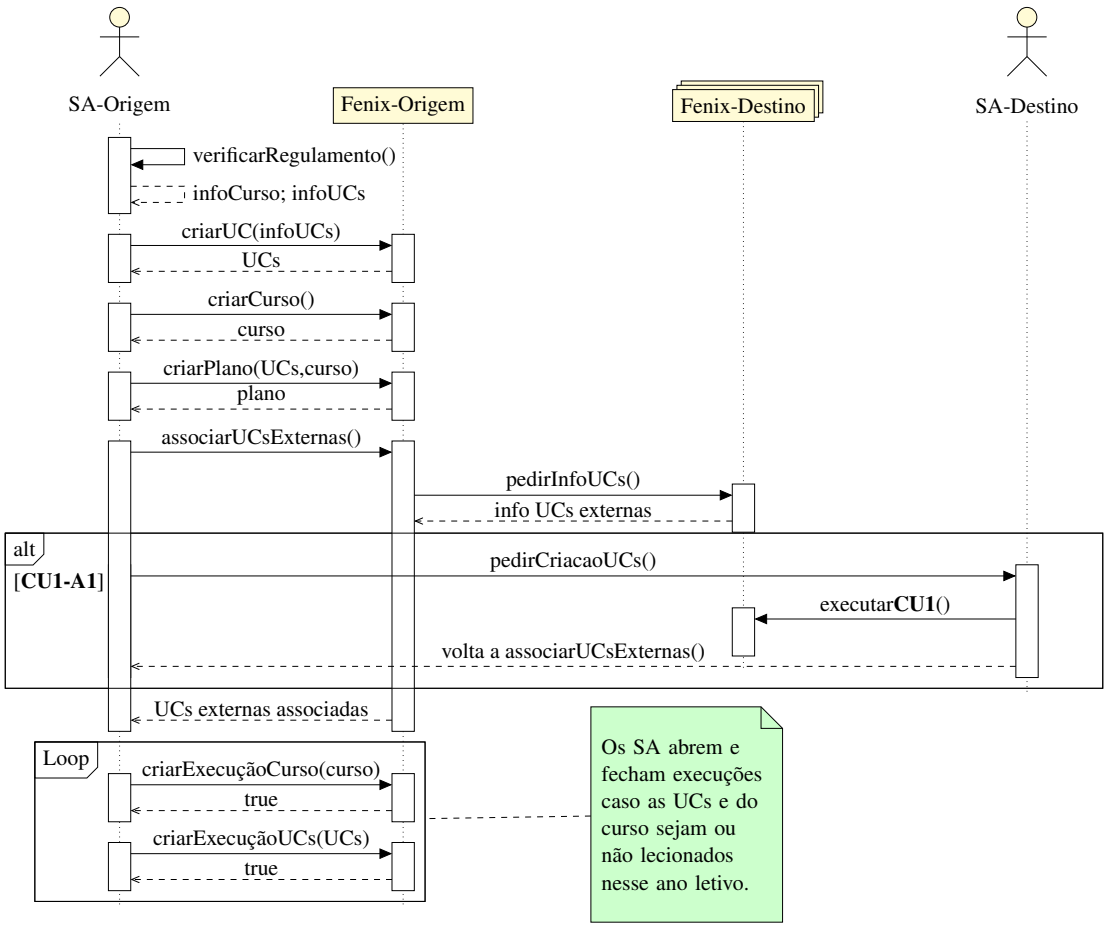

Figura 4.2: Diagrama de sequência do caso de uso de criação do curso na escola de origem.

CU3 - Matrícula do aluno

Nome: Matrícula do aluno

Ator Principal: Aluno

Período: Durante o período de matrículas

Pré-Condições:
- Os SA-Origem importaram o ficheiro de colocados no Fenix.
- O aluno está colocado no curso.
- O aluno não está matriculado no curso.

Pós-Condições:
- O aluno fica matriculado no curso.

Cenário Principal de Sucesso:

1- O aluno inicia o processo de matrícula.

2- O aluno preenche o inquérito de matrícula.

3- O aluno prossegue para a inscrição no caso de uso (**CU4**).

Diagrama de sequência: Figura 4.3

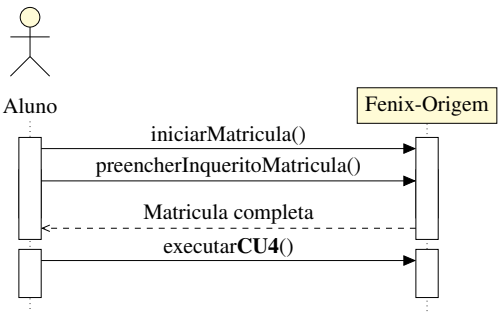

Figura 4.3: Diagrama de sequência do caso de uso de matrícula do aluno.

CU4 - Inscrição do aluno
Nome: Inscrição do aluno
Ator Principal: Aluno
Atores Secundários: SA-Origem, SA-Destino
Período: Durante o período de inscrições
Pré-Condições: • O aluno está matriculado.
Pós-Condições: • O aluno fica inscrito no ano corrente no curso e nas UCs.
Cenário Principal de Sucesso:

1- O aluno inicia o processo de inscrição.
2- O Fenix-Origem pede às instâncias Fenix-Destino as UCs lecionadas do curso que estão em execução no presente ano letivo.
3- O Fenix-Origem apresenta ao aluno as UCs do seu curso.
4- O aluno inscreve-se nas UCs que pretende realizar.
5- O Fenix-Origem comunica as inscrições das UCs às instâncias Fenix onde as UCs são lecionadas.
6- O Fenix-Destino inscreve o aluno nas UCs.
7- O Fenix-Destino comunica os turnos das UCs às quais o aluno se inscreveu ao Fenix-Origem.
8- O aluno inscreve-se nos turnos que pretende.
9- O aluno finaliza a inscrição.
10- O Fenix-Origem comunica as inscrições dos turnos às instâncias Fenix onde as UCs são lecionadas.
11- Os SA-Origem confirmam a inscrição do aluno.

Cenários Alternativos:

CU4-A1: 0a - Reingresso de um aluno de um curso em gestão rotativa que interrompeu os estudos

1 - O aluno deve realizar a matricula, , executando o caso de uso (**CU3**), no Fenix da escola gere o curso no ano letivo corrente.
2 - O aluno continua o processo de inscrição no ponto 1.

CU4-A2: 9a - Caso não existam vagas nos turnos de uma das UCs.

1 - O aluno volta atrás no processo de inscrição.

2 - O Fenix-Origem comunica a anulação das inscrições nas UCs às quais o aluno se estava a inscrever.

3 - O fluxo volta ao ponto 4.

CU4-A3: 9b - Caso existam ao mesmo tempo, mais alunos a confirmar um turno do que vagas para o mesmo.

1 - O Fenix-Origem comunica a anulação das inscrições nas UCs e nos turnos aos quais o aluno se estava a inscrever.

2 - O fluxo volta ao ponto 4.

Diagrama de sequência: Figura 4.4

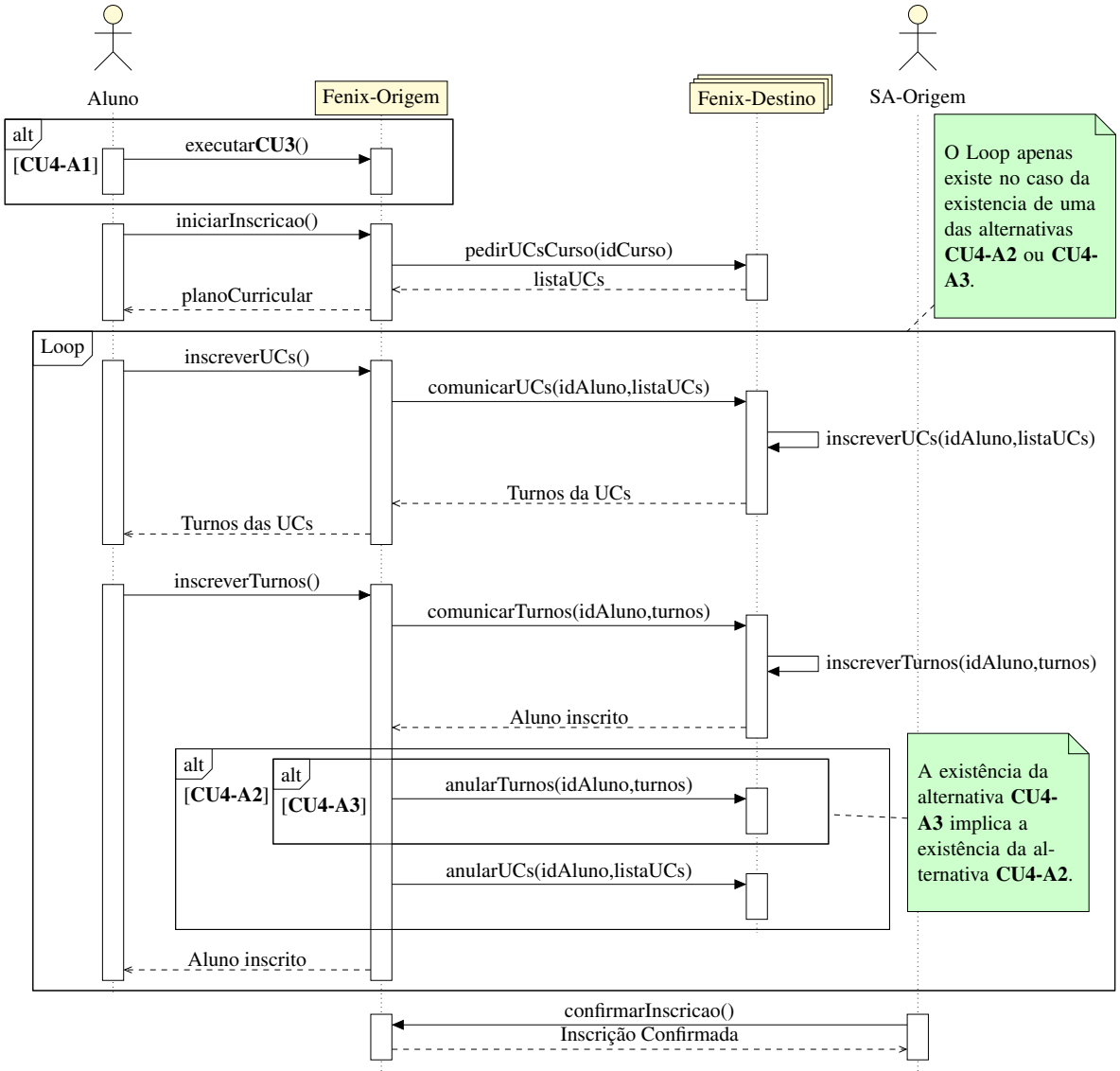

Figura 4.4: Diagrama de sequência do caso de uso de inscrição do aluno.

CU5 - Alteração das inscrições
Nome: Alteração das inscrições
Ator Principal: Aluno
Atores Secundários: SA-Origem
Período: Durante os períodos de alteração de inscrições
Pré-Condições:
- O aluno está matriculado.
- O aluno está inscrito no curso no ano letivo corrente.

Pós-Condições:
- O aluno fica com a sua inscrição alterada.

Cenário Principal de Sucesso:

1- O aluno pede aos SA-Origem para fazer alteração da inscrição.
2- O aluno escolhe as UCs que pretende alterar.
3- Os SA-Origem alteram as UCs no Fenix-Origem.
4- O Fenix-Origem comunica às instâncias Fenix-Destino responsáveis pelas UCs, as alterações efetuadas.

Diagrama de sequência: Figura 4.5

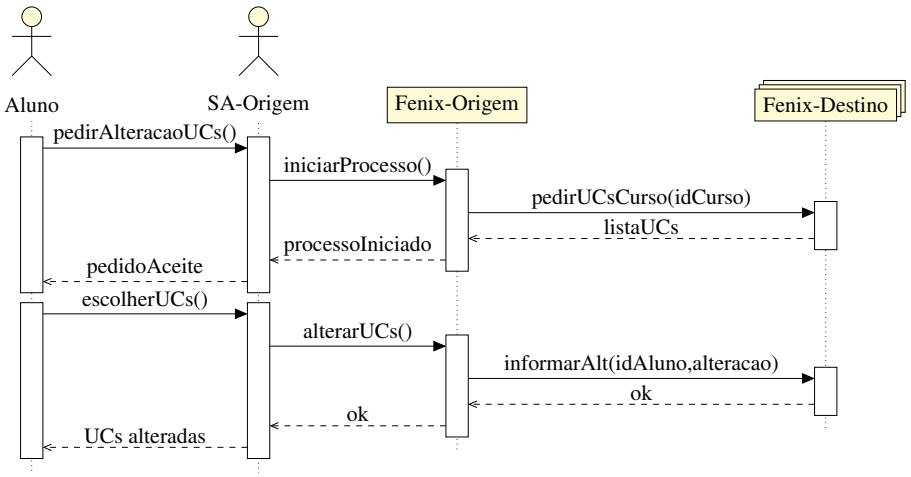

Figura 4.5: Diagrama de sequência do caso de uso de alteração das inscrições.

CU6 - Transferência de escola de um aluno (sem alteração de curso)
Nome: Transferência de escola de um aluno (sem alteração de curso)
Ator Principal: SA-Origem
Período: Preparação do ano letivo.
Pré-Condições:
- O aluno transitou de ano.

Pós-Condições:
- O aluno fica inscrito na escola de destino
- A escola de origem passa a ser a escola de destino.
- A escola de destino passa a ser a escola de origem.

Cenário Principal de Sucesso:

1- Os SA-Origem informam o Fenix-Origem que o aluno foi transferido para

outra escola do curso.

2- O Fenix-Origem envia o processo do aluno para o Fenix-Destino.

3- As instâncias Fenix trocam, a de origem passa a ser de destino e vice-versa.

Diagrama de sequência: Figura 4.6

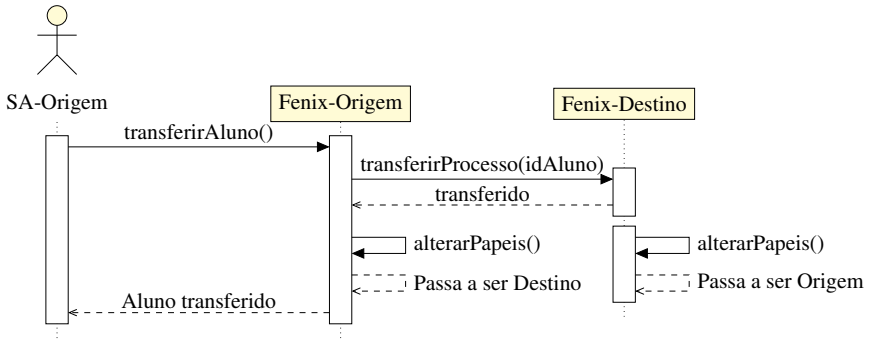

Figura 4.6: Diagrama de sequência do caso de uso de transferência de escola de um aluno (sem alteração de curso).

CU7 - Inscrição a avaliações

Nome: Inscrição a avaliações

Ator Principal: Aluno

Período: Durante a época de inscrição a avaliações

Pré-Condições: • O aluno está inscrito às UCs.

Pós-Condições: • O aluno fica inscrito às avaliações das UCs.

Cenário Principal de Sucesso:

1- O aluno acede à aba de inscrição a avaliações na sua instância Fenix.

2- O Fenix-Origem pede às instâncias Fenix-Destino, onde o aluno está a realizar UCs, a lista de avaliações que o aluno pode realizar.

3- O Fenix-Origem apresenta ao aluno as avaliações às quais se pode inscrever.

4- O aluno faz a inscrição nos momentos de avaliação das UCs.

5- O Fenix-Origem comunica às instâncias Fenix-Destino responsáveis pelas avaliações, as inscrições que o aluno realizou.

Diagrama de sequência: Figura 4.7

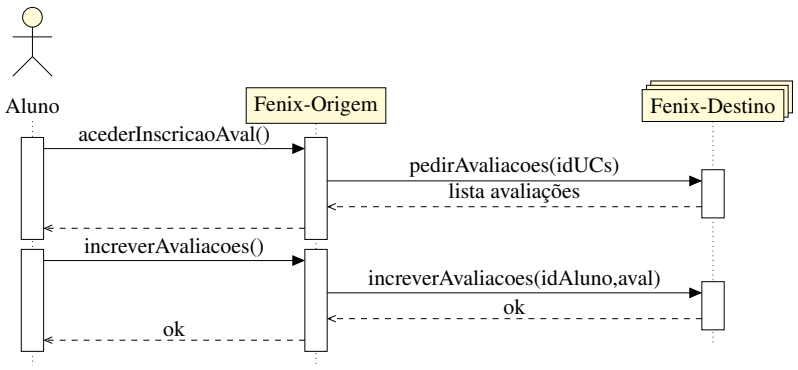

Figura 4.7: Diagrama de sequência do caso de uso de inscrição a avaliações.

CU8 - Lançar a pauta de uma UC com alunos de cursos partilhados
Nome: Lançar a pauta de uma UC com alunos de cursos partilhados
Ator Principal: Professor
Atores Secundários: SA-Destino
Período: Durante as épocas de avaliação
Pré-Condições: • O professor é responsável pela à UC.
Pós-Condições: • A pauta foi lançada na escola onde é lecionada a UC.
Cenário Principal de Sucesso:

1- O professor lança a pauta da sua UC, na instância Fenix da sua escola.
2- Os SA verificam a pauta.

Diagrama de sequência: Figura 4.8

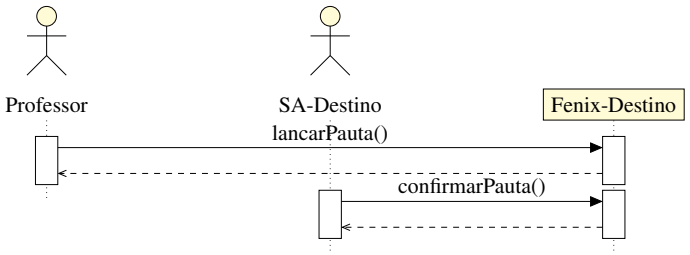

Figura 4.8: Diagrama de sequência do caso de uso de lançamento da pauta de uma UC com alunos de cursos partilhados.

CU9 - Visualização do currículo
Nome: Visualização do currículo
Ator Principal: Aluno
Período: Durante todo o ano.
Pré-Condições: • O aluno tem acesso ao Fenix.
Pós-Condições: • O aluno visualiza o seu currículo.

Nota: • O ator principal pode ser aluno, ex-aluno ou os SA, para este caso é utilizado o aluno como exemplo, sendo que o fluxo é igual para os restantes casos.

Cenário Principal de Sucesso:

1- O aluno acede ao seu currículo, na sua instância Fenix.

2- O Fenix-Origem pede às instâncias Fenix-Destino onde o aluno realizou UCs as informações relativas ao currículo, nota, turmas/turnos, peso, ECTS, época de avaliação, ano de realização e período letivo.

3- O Fenix-Origem apresenta ao aluno o seu currículo agregado.

Diagrama de sequência: Figura 4.9

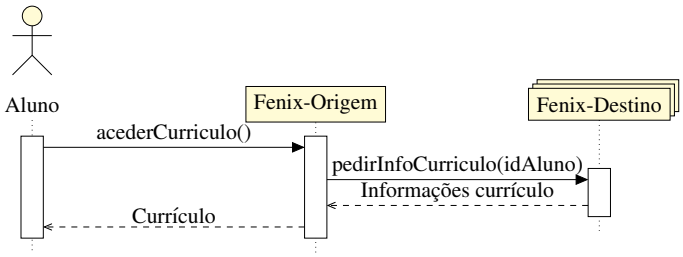

Figura 4.9: Diagrama de sequência do caso de uso de visualização do currículo.

CU10 - Inquéritos às UCs

Nome: Inquéritos às UCs

Ator Principal: Aluno

Período: Período de Inquéritos

Pré-Condições: • O aluno está inscrito às UCs.

Pós-Condições: • O aluno realizou o inquérito às UCs.

Cenário Principal de Sucesso:

1- O aluno acede à aba de inquéritos na sua instância Fenix.

2- O Fenix-Origem pede às instâncias Fenix-Destino onde o aluno realizou UCs, os inquéritos por responder.

3- O Fenix-Origem apresenta os inquéritos ao aluno.

4- O aluno responde aos inquéritos das UCs que realizou.

5- O Fenix-Origem envia os inquéritos para as respetivas instâncias Fenix-Destino.

Diagrama de sequência: Figura 4.10

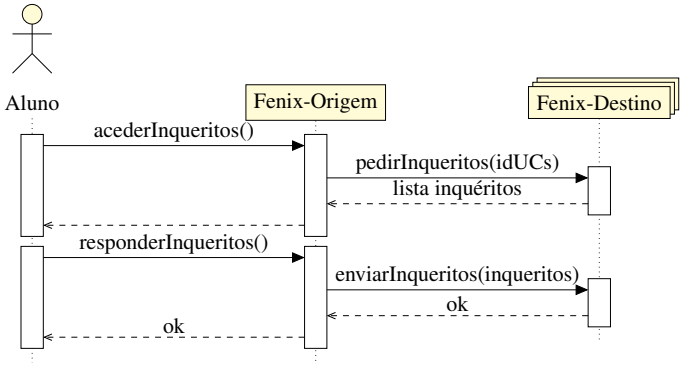

Figura 4.10: Diagrama de sequência do caso de uso de inquéritos às UCs.

CU11 - Pedido de documentos académicos
Nome: Pedido de documentos académicos
Ator Principal: Aluno
Período: Durante todo o ano
Pré-Condições: • O aluno tem acesso ao Fenix.
Pós-Condições: • O aluno recebe o documento pedido.
Cenário Principal de Sucesso:

1- O aluno realiza o pedido de documentos académicos.
2- O Fenix-Origem pede a informação necessária para a criação do documento, às instâncias Fenix-Destino responsáveis por essa mesma informação.
3- O Fenix-Origem cria o documento.
4- O Fenix-Origem disponibiliza o documento ao aluno.

Cenários Alternativos:

CU11-A1: 3a - O documento implica o pagamento de emolumentos

1 - O aluno deve realizar o pagamento do emolumento.

Diagrama de sequência: Figura 4.11

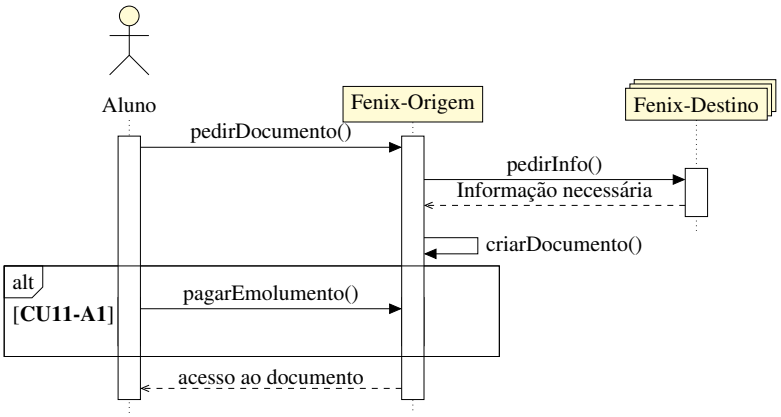

Figura 4.11: Diagrama de sequência do caso de uso de pedido de documentos académicos.

CU12 - Visualizar o horário do aluno
Nome: Visualizar o horário do aluno
Ator Principal: Aluno
Período: Ao longo do ano letivo
Pré-Condições: • O aluno está inscrito neste ano letivo.
Cenário Principal de Sucesso:

1- O aluno acede ao seu horário na sua instância Fenix.
2- O Fenix-Origem pede às instâncias Fenix-Destino onde o aluno está inscrito a UCs, o horário das UCs e turnos em que o aluno está inscrito.
3- O Fenix-Origem apresenta ao aluno o seu horário agregado.

Diagrama de sequência: Figura 4.12

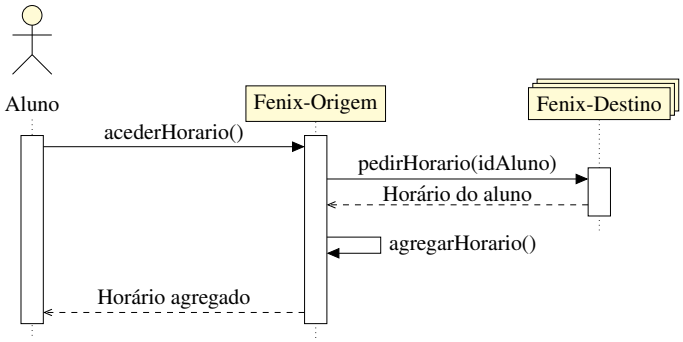

Figura 4.12: Diagrama de sequência do caso de uso de visualização do horário do aluno.

CU13 - Visualizar o horário de um semestre do curso
Nome: Visualizar o horário de um semestre do curso
Ator Principal: SA
Período: Ao longo do ano letivo
Pré-Condições: • Existe execução do curso e das UCs para o ano corrente.
Cenário Principal de Sucesso:

1- Os SA acedem ao horário de um semestre do curso no Fenix que gere o curso.
2- Por cada UC externa, o Fenix-Origem pede à instância responsável pela UC os seus turnos.
3- O Fenix-Origem apresenta horário do curso para o semestre escolhido.

Diagrama de sequência: Figura 4.13

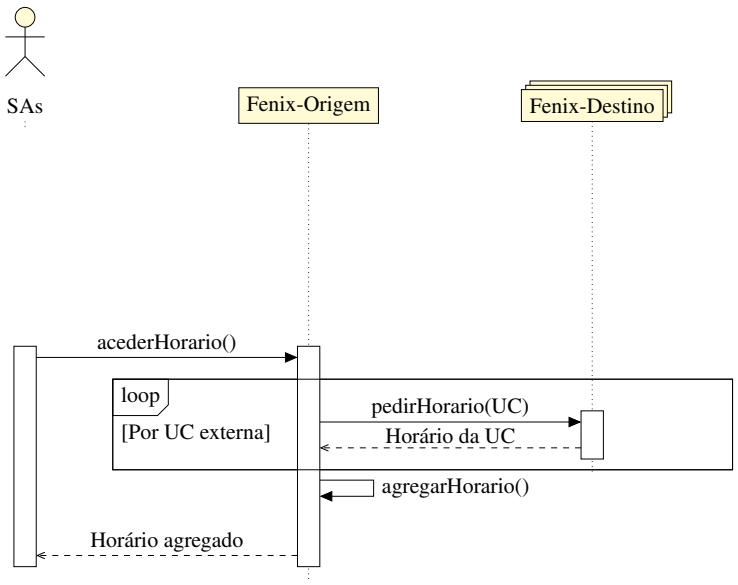

Figura 4.13: Diagrama de sequência do caso de uso de visulização do horário de um semestre do curso.

CU14 - Visualizar a informação da página pública do curso e das UCs

Nome: Visualizar a informação da página pública do curso e das UCs

Ator Principal: Aluno

Período: Durante todo o ano

Pré-Condições:
- O curso está criado.
- As UCs estão criadas e associadas ao curso.

Nota:
- Este caso de uso pode ser realizado por qualquer utilizador, para este caso é utilizado o aluno, sendo que o fluxo é igual para os restantes.

Cenário Principal de Sucesso:

1- O aluno acede a instância Fenix responsável pelo curso.

2- O Fenix-Origem pede às instâncias Fenix-Destino responsáveis pelas UCs do curso, a informação sobre as mesmas.

3- O aluno visualiza a informação do curso e das UCs.

Diagrama de sequência: Figura 4.14

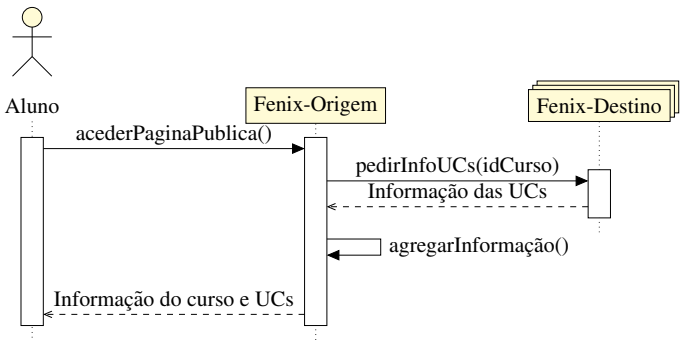

Figura 4.14: Diagrama de sequência do caso de uso de visualização da informação da página pública do curso e das UCs.

CU15 - Visualizar a conta corrente do aluno

Nome: Visualizar a conta corrente do aluno

Ator Principal: Aluno

Período: Ao longo do ano letivo

Nota:
- Tipicamente o aluno apenas tem a necessidade de aceder a diferentes Fenix para ver emolumentos específicos ou propinas antigas no caos de ter sido transferido.
- Este caso é mais abrangente que o caso de cursos partilhados, abrange também alunos que realizam cursos ou UCs em diferentes escolas, por esta razão não se encontra dentro do âmbito deste projeto.

Cenário Principal de Sucesso:

1- O aluno visualiza a sua conta corrente na sua escola de origem.

2- O aluno visualiza a sua conta corrente nas suas escolas de destino.

Diagrama de sequência: Figura 4.15

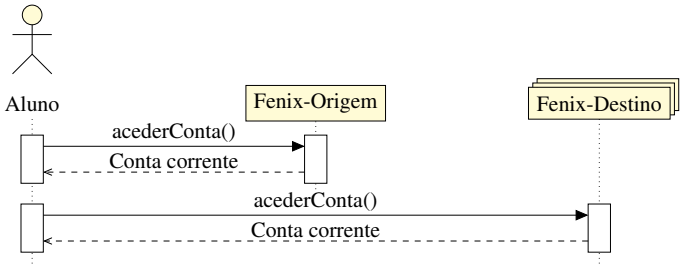

Figura 4.15: Diagrama de sequência do caso de uso de visualização da conta corrente do aluno.

CU16 - Candidatura a proposta de formação avançada

Nome: Candidatura a proposta de formação avançada

Ator Principal: Aluno

Período: Período de candidaturas a propostas de formação avançada

Pré-Condições:
- O aluno está em condições de realizar a formação avançada.

Pós-Condições: • O aluno candidatou-se a proposta de formação avançada.

Nota: • Devido às diferenças dos períodos de candidaturas de formação avançada nas diferentes escolas, o processo de seriação continua com bloqueios na confirmação do aluno para a sua nomeação para a proposta.

Cenário Principal de Sucesso:

1- O aluno inicia o candidatura a formação avançada.

2- O Fenix-Origem pede às restantes instâncias Fenix-Destino as propostas de formação avançada para o curso.

3- O Fenix-Origem apresenta ao aluno as propostas existentes.

4- O aluno escolhe as suas propostas preferenciais.

5- O Fenix-Origem comunica às instâncias Fenix-Destino responsáveis pelas proposta escolhidas pelo aluno a sua escolha, enviando o processo do aluno para que este possa ser seriado.

Diagrama de sequência: Figura 4.16

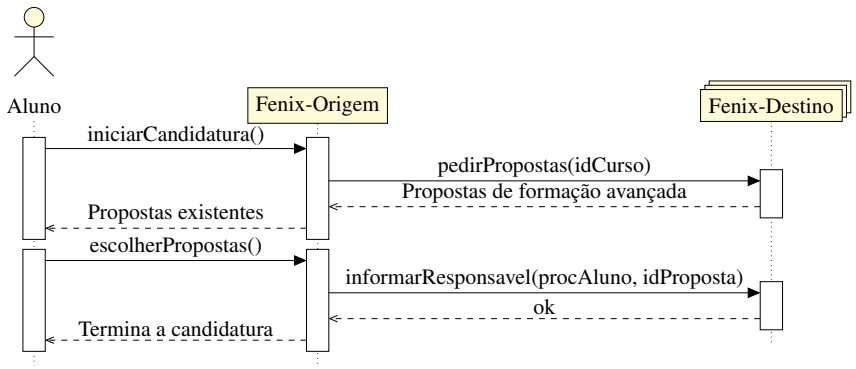

Figura 4.16: Diagrama de sequência do caso de uso de candidatura a proposta de formação avançada.

CU17 - Visualização do processo de formação avançada

Nome: Visualização do processo de formação avançada

Ator Principal: Aluno

Período: Durante o período de formação avançada

Pré-Condições: • O aluno está a realizar formação avançada.

Cenário Principal de Sucesso:

1- O aluno acede ao seu processo de formação avançada.

2- O Fenix-Origem apresenta ao aluno a informação sobre o seu processo.

3- O aluno faz *upload* dos documentos da sua formação avançada.

Cenários Alternativos:

CU15-A1: 1a - A proposta não pertence à escola de origem do aluno.

1 - O Fenix.Origem pede ao Fenix-Destino responsável pelo processo a informação

do mesmo.

2 - O fluxo continua do ponto 2 ao 3.

3 - O Fenix-Origem envia para o Fenix-Destino responsável os documentos do aluno.

Diagrama de sequência: Figura 4.17

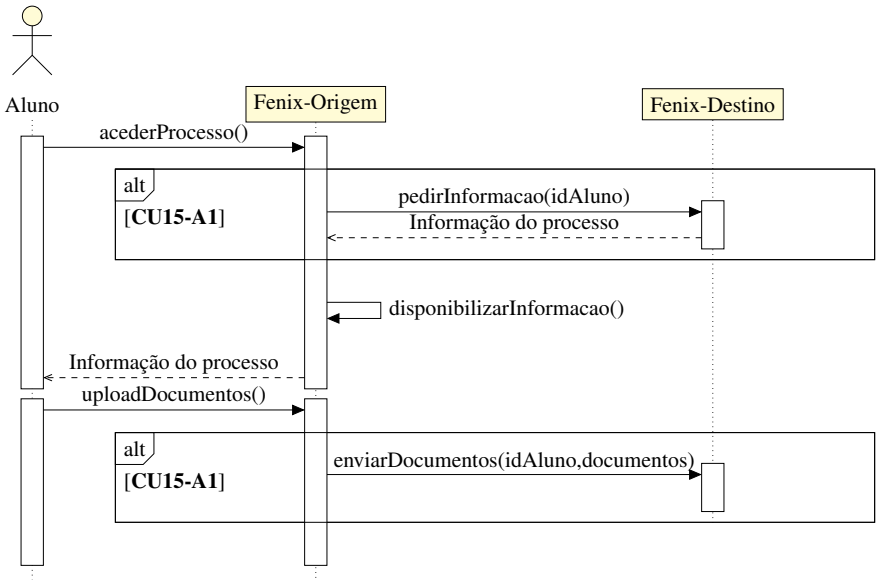

Figura 4.17: Diagrama de sequência do caso de uso de visualização do processo de formação avançada.

CU18 - Candidatura de mobilidade *outgoing*

Nome: Candidatura de mobilidade *outgoing*

Ator Principal: Aluno

Período: Durante o período de candidaturas de mobilidade *outgoing*

Pré-Condições: • O período de candidaturas está aberto.

Pós-Condições: • O aluno fica nomeado para mobilidade *outgoing*.

Nota: • Devido às diferenças dos períodos de candidaturas de mobilidade *outgoing* nas diferentes escolas, o processo de seriação continua com bloqueios na confirmação do aluno para a sua nomeação para o acordo.

Cenário Principal de Sucesso:

1- O aluno cria um processo de candidatura *outgoing* no seu Fenix.

2- O Fenix-Origem pede às restantes instâncias Fenix-Destino os acordos possíveis para o curso do aluno.

3- O Fenix-Origem apresenta ao aluno os acordos existentes.

4- O aluno escolhe os seus acordos preferenciais.

5- O Fenix-Origem envia um pedido de avaliação da candidatura para às instâncias Fenix-Destino das escolas responsáveis pelos acordos.

6- O coordenador Erasmus/DREI da escola avalia a candidatura.
7- O aluno é seriado.
8- O aluno escolhe uma das opções nas quais ficou colocado.
9- O aluno cria as correspondências entre as UCs.
10- O Fenix-Origem envia um pedido de avaliação das correspondências para a instância Fenix-Destino da escola responsável por cada UC.
11- Os coordenadores Erasmus/DREI das escolas responsáveis pelas UCs avaliam as correspondências.

Cenários Alternativos:

CU15-A1: 9a - Não existem correspondências necessárias entre as UCs.

1 - O processo volta ao ponto 7.

Diagrama de sequência: Figura 4.18

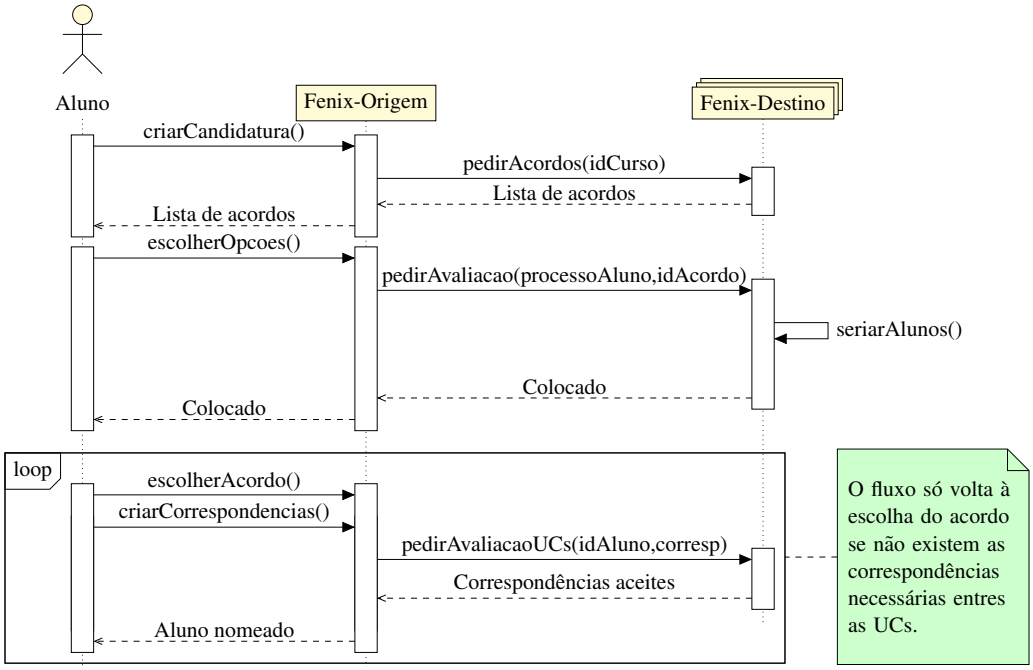

Figura 4.18: Diagrama de sequência do caso de uso de candidatura de mobilidade *outgoing*.

CU19 - Criar relatório para RAIDES

Nome: Criar relatório para RAIDES

Ator Principal: SA-Origem

Período: Em dois momentos no ano letivo, o primeiro a 31 de dezembro e o segundo a 31 de março.

Pré-Condições:
- O curso faz parte do âmbito do relatório.
- Os alunos inscritos/diplomados fazem parte do âmbito do relatório.

Pós-Condições:
- Os documentos para o relatório são gerados.

Cenário Principal de Sucesso:

1- Os SA atualizam a configuração dos RAIDES.
2- Os SA criam um pedido de criação de relatório.
3- Os SA fazem download dos relatórios.
4- Os SA verificam os dados.
5- Os SA entregam o relatório na PRIES.

Cenários Alternativos:

CU15-A1: 3a - No caso dos alunos serem divididos pelas diferentes escolas do curso.

1 - Os SA dividem o documento gerado pelas diferentes escolas do curso.
2 - Os SA das restantes escolas adicionam esta informação aos seus documentos.
3 - O fluxo continua no ponto 4, sendo executado por cada uma das escolas.

Diagrama de sequência: Figura 4.19

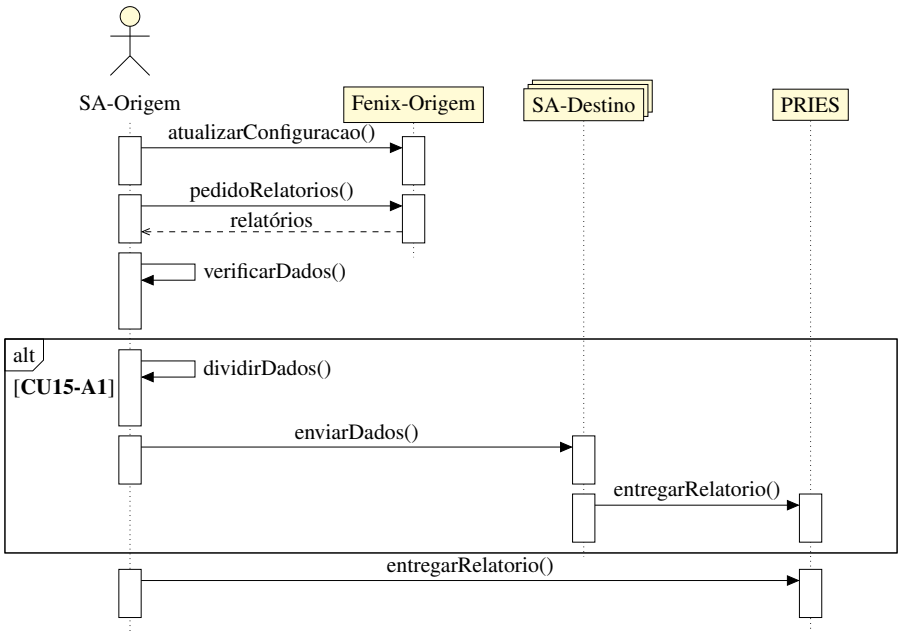

Figura 4.19: Diagrama de sequência do caso de uso de criar relatório para RAIDES.

CU20 - Candidatura a bolsas SAS

Nome: Candidatura a bolsas SAS

Ator Principal: Aluno

Atores Secundários: SA-Origem

Período: Inicia antes das matriculas/inscrições e termina no início do primeiro semestre.

Pré-Condições: • O aluno está colocado no curso ou matriculado no curso.

Pós-Condições: • O aluno fica com o processo de candidatura completo.

Cenário Principal de Sucesso:

1- O aluno realiza a candidatura a bolsa através da DGES.
2- O aluno realiza o caso de uso **CU3** e o **CU4**.
3- A DGES envia a informação/pedido para o Fenix da escola responsável pelo curso.
4- O Fenix-Origem pede às restantes instâncias Fenix-Destino a informação necessária para complementar a informação existente, como por exemplo a média do aluno.
5- Os SA preenchem os dados em falta que tem de ser inseridos/confirmados manualmente.
6- Os SA reenviam os dados para a DGES.
7- A DGES envia atualizações para o Fenix responsável pelo curso, o fluxo volta ao ponto 4.
8- A DGES envia a informação de bolsa deferida para o Fenix responsável pelo curso.
9- O Fenix lança o estatuto de bolseiro SAS ao aluno.

Diagrama de sequência: Figura 4.20

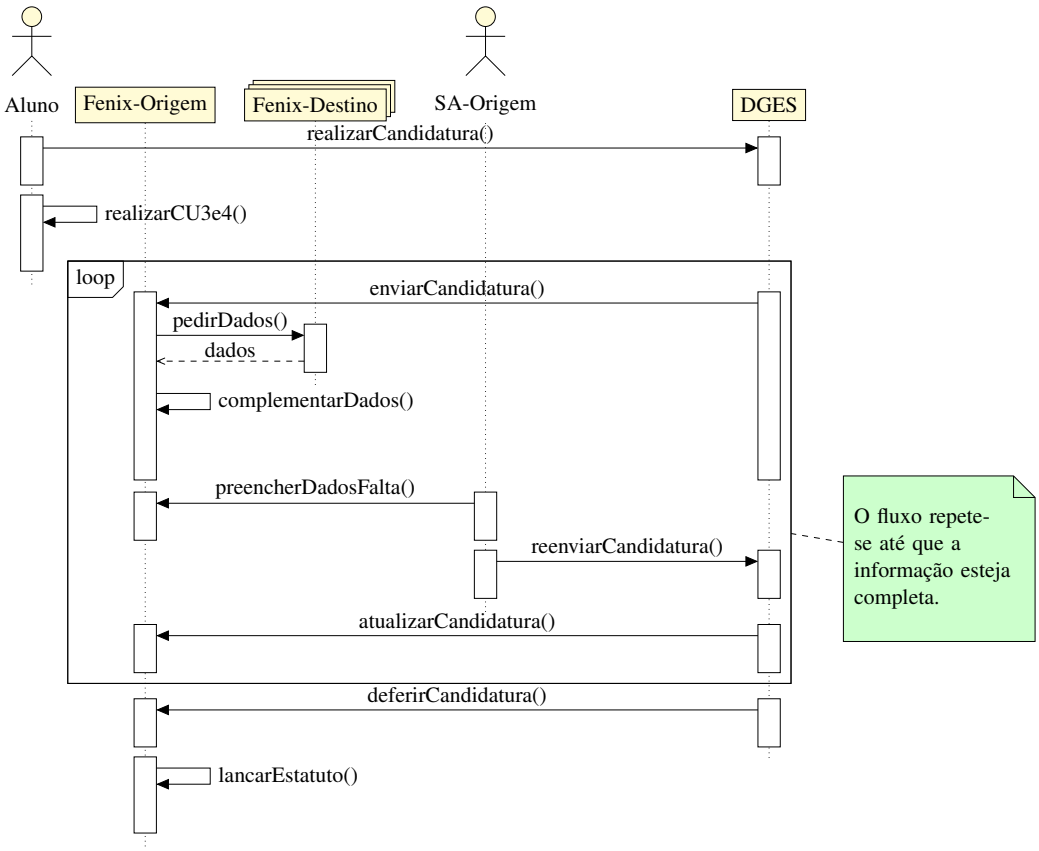

Figura 4.20: Diagrama de sequência do caso de uso de candidatura a bolsas SAS

CU21 - Acerto de contas
Nome: Acerto de contas
Ator Principal: Serviços Financeiros
Período: No final do ano letivo.
Descrição:

No final do ano letivo, as escolas realizam o acerto de contas, com movimentos contabilísticos entre os serviços financeiros de cada uma das escolas. Este acerto de contas é realizado fora do Fenix e pode ser feito de duas maneiras. Caso o acerto de contas seja efetuado de acordo com a percentagem de UCs disponibilizadas no plano, a funcionalidade do Fenix não apresenta vantagens adicionais. No caso do acerto de contas ser feito com base no número de alunos inscritos nas UCs de cada uma das escolas, a funcionalidade do Fenix pode disponibilizar o cálculo dos valores do acerto.

4.3 Requisitos não-funcionais

Os principais requisitos não-funcionais definidos para o trabalho realizado no contexto deste projeto são:

- **Desempenho** - A gestão dos cursos partilhados deve ser transparente para os alunos, de forma a que não seja perceptível o acesso a outras instâncias.
- **Evitar a replicação de dados** - A inexistência de réplicas evita dois tipos de problemas: a coerência dos dados e a gestão da propriedade dos dados.

A solução final para gestão de cursos partilhados no Fenix deve ter em conta também os seguintes requisitos:

- **Usabilidade** - A gestão dos cursos partilhados deve ser transparente para os alunos, de forma a que eles apenas precisem de utilizar uma instância Fenix.
- **Flexibilidade** - Devido à autonomia das escolas e as diferenças de calendário, nomenclatura, entre outros, bem como as diferentes tipologias de cursos partilhados. As soluções a estudar devem ser flexíveis para responder da melhor forma a estas dificuldades.
- **Normalização** - A solução encontrada deve manter as normas utilizadas na restante aplicação.

4.4 Sumário

Neste capítulo foram apresentados os objetivos, os pressupostos, os casos de uso e os requisitos não-funcionais a ter em conta para a criação da solução que visa integrar a gestão de cursos partilhados no sistema Fenix. No capítulo seguinte, estes requisitos são utilizados para justificar as escolhas ao nível da arquitetura, desenho e implementação da solução proposta.

Capítulo 5

Arquitetura, desenho e implementação

Este capítulo apresenta as alternativas estudadas do ponto de vista da arquitetura e do desenho da solução proposta para integrar a gestão de cursos partilhados no sistema Fenix, bem como alguns detalhes da sua implementação.

5.1 Arquitetura

O projeto de cursos partilhados no sistema Fenix transforma as várias instâncias isoladas do Fenix num sistema distribuído que partilha dados.

O teorema do *CAP* de Eric Brewer diz que um sistema em rede que partilhe dados apenas pode garantir duas das propriedades seguintes: **Coerência**, um objeto partilhado e a sua réplica aparecem no sistema como uma única cópia; **Disponibilidade**, existe a garantia de resposta, num pedido feito ao sistema, mesmo que não seja a versão mais recente; **Particionamento**, o sistema continua a funcionar apesar da perda de mensagens. Este teorema baseia-se num *trade-off* entre a coerência dos dados e a sua disponibilidade[12]. Neste trabalho, pretende-se minimizar os problemas de gestão de coerência evitando a utilização de réplicas dos dados.

De seguida são discutidos o estilo arquitetural e a arquitetura de sistema considerados para o projeto, tendo por base os conceitos teóricos de sistemas distribuídos de Steen e Tanenbaum [12].

O estilo arquitetural de um sistema é definido em termos dos seus componentes, a forma como estes se interligam e trocam dados. Neste projeto, cada instância do Fenix constitui um componente, os quais são estruturados por camadas, seguindo o paradigma orientado a objetos (ver detalhes no capítulo 2).

O estilo arquitetural escolhido para o projeto foi definido pelos requisitos que pretendem manter as várias instâncias do Fenix autónomas e evitar a duplicação de informação para minimizar os problemas de sincronização e coerência dos dados.

Tendo em conta o estilo arquitetural de cada instância do Fenix e o requisitos, optou-se pelo estilo de arquitetura baseado em objetos, mais propriamente o de arquitetura baseada

em serviços, através do qual o acesso aos objectos pelas diversas instâncias é assegurado através de invocações remotas. De forma a reduzir do número de mensagens trocadas entre as diversas instâncias, optou-se por criar um sistema de cache onde cada instância guardará os dados dos objetos remotos durante um período de tempo predefinido.

Em alternativa, foi também considerado o estilo arquitetural baseado em eventos, o qual permite uma maior escalabilidade ao sistema, visto que a adição de novos nós pode passar apenas pela inserção de mais subscritores, o que não obriga a um esforço complementar dos nós já existentes. As vantagens deste estilo são permitir que as instâncias comuniquem sem estarem ativas ao mesmo tempo e a redução do número de mensagens trocadas, porque neste caso, só existe troca de mensagens quando o objeto é alterado. No entanto, este estilo requer a replicação dos dados e, consequentemente, o suporte a mecanismos que assegurem a sua sincronização.

As escritas sobre objetos remotos levanta um problema de coerência. Visto que o Fenix utiliza a JVSTM na sua *framework*, optou-se por resolver este problema ao nível da camada aplicacional da seguinte forma: caso uma transação nos objetos remotos seja abortada pela JVSTM da *framework* Fenix, a escrita deve conter procedimentos que garantam a possibilidade de desfazer alterações que foram realizadas, através de invocações remotas. Este problema poderia ainda ser abordado ao nível da camada de persistência, tornando a JVSTM distribuída. Sérgio Fernandes, na sua tese de doutoramento [3], apresenta as alterações necessárias para tornar a JVSTM distribuída. Estas pressupõem a utilização de um repositório de dados transacional, que garanta que os dados estão disponíveis em todos os nós. As alterações são: Após um *commit* bem sucedido, o relógio global fornece um número de versão único a todos os nós. Para validar um *commit* é enviada uma mensagem com a identificação das versões e não os valores. As mensagens devem ser enviadas dentro de um *lock* global. As transações de escrita, depois de entrarem num *lock* global, devem garantir que as mensagens relativas a *commits* anteriores já foram processadas. O mesmo deve ser garantido antes de uma nova transação de leitura, garantido assim que a transação começa com a versão mais recente dos dados.

Em relação à arquitetura do sistema, Steen e Tanenbaum distinguem três diferentes arquiteturas: centralizadas, descentralizadas e híbridas. As descentralizadas podem ser estruturadas ou não estruturadas. Neste projeto optou-se pela arquitetura descentralizada não estruturada. Esta opção é justificada pelo facto das instâncias serem autónomas e precisarem apenas de comunicar com as instâncias com as quais partilham um ou mais cursos.

Na Figura 5.1 é ilustrada a arquitetura do projeto, onde cada elipse representa uma instância Fenix autónoma e cada seta representa a invocação remota suportada por serviços *web*. Na imagem, as instâncias de onde partem as invocações remotas são instâncias que gerem um ou mais cursos partilhados com as instâncias de destino destas invocações.

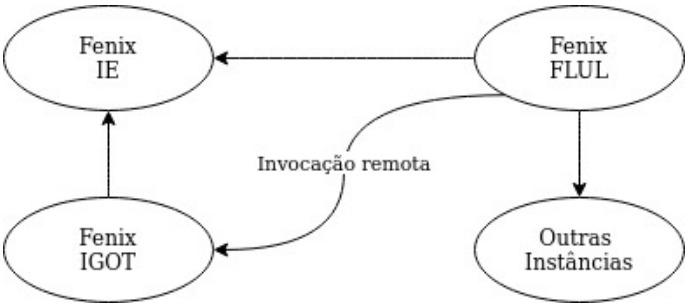

Figura 5.1: Arquitetura do sistema distribuído do projeto, as elipses representam as instâncias Fenix e as setas representam as invocações remotas.

5.2 Desenho

Nesta seção são apresentadas as decisões de desenho ao nível das alterações do domínio do sistema Fenix e da configuração de um curso partilhado. No entanto, devido à extensão do domínio, são utilizados como exemplos os casos de uso de criação de um curso na escola de origem (**CU2**) e a visualização do horário de um semestre de um curso (**CU13**).

Para que o projeto possa ser suportado, é necessário que a instância que gere o curso consiga distinguir entidades locais de entidades remotas (que podem ser acedidas através de invocações remotas) e suportar um comportamento específico para cada um dos casos. Desta forma, optou-se pela utilização do mecanismo de herança do Java, em detrimento do mecanismo de composição e da utilização de aspetos. Isto é justificado pela necessidade das entidades remotas serem utilizadas pelo sistema Fenix como locais, mas com comportamentos específicos. O mecanismo de composição obrigaria a uma refratorização das entidades. Para a utilização de aspetos, seria necessário utilizar atributos dos aspetos, facto que entrava em conflito com a *framework* Fenix já que estes não seriam persistidos pela *framework* (a *framework* só persiste os dados que estão descritos na DML).

A Figura 5.2 apresenta as classes de domínio existentes no Fenix referentes aos casos de uso, as extensões criadas e as classes que permitem identificar a que escola pertencem as entidades remotas. No Apêndice A é apresentado um modelo de classes mais detalhado, envolvendo mais casos de uso. A classe *ExternalFenixULExecutionCourse*, uma extensão da classe *ExecutionCourse*, mantém o comportamento da segunda, à exceção dos métodos que dão acesso às entidades *LessonPlanning*, *BibliographicReference*, *Professorship*, *Summary* e *CourseLoad*. É nestes métodos que são feitas as invocações remotas às instâncias Fenix que são responsáveis por estas entidades. Para que estes serviços sejam realizados é necessário que a entidade consiga identificar a que instância pertence. A classe *ExternalInstituitionCommunication* faz a identificação da instituição remota e está associada à *CompetenceCourse*, a partir da qual as entidades identificam a instituição a que pertencem.

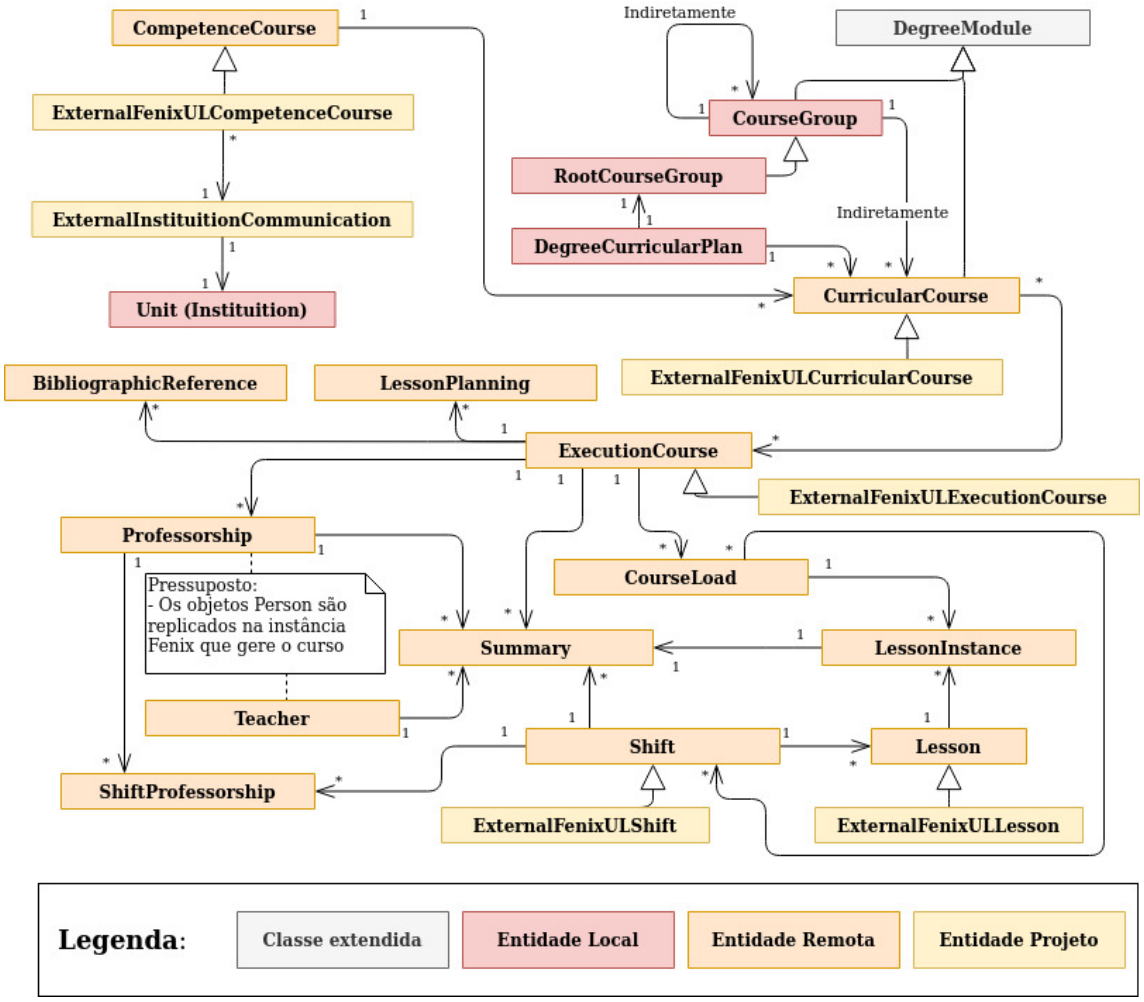

Figura 5.2: Diagrama de classes externas dos cursos partilhados

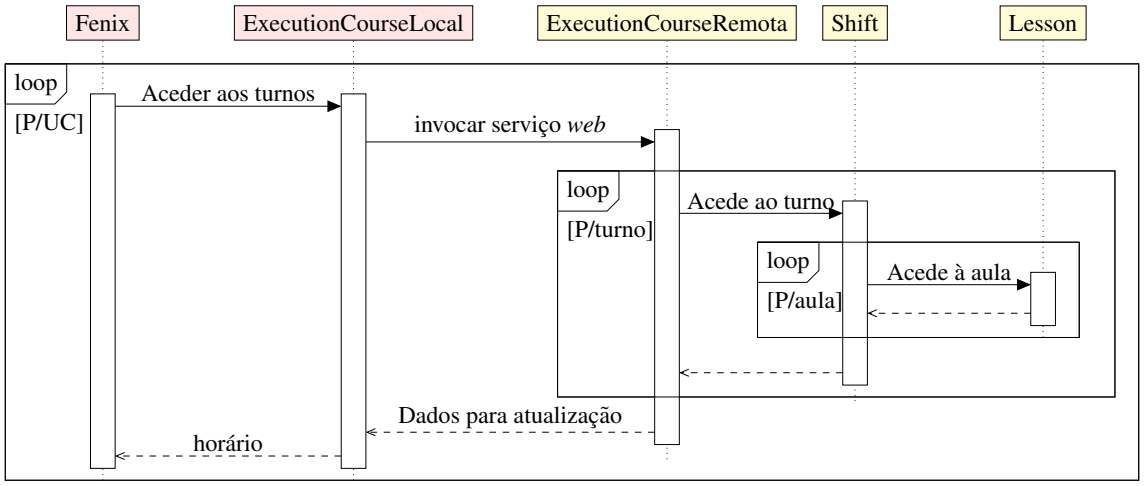

Figura 5.3: Diagrama de sequência de um pedido de serviço *web*. A vermelho estão representadas as classes locais e a amarelo as classes remotas.

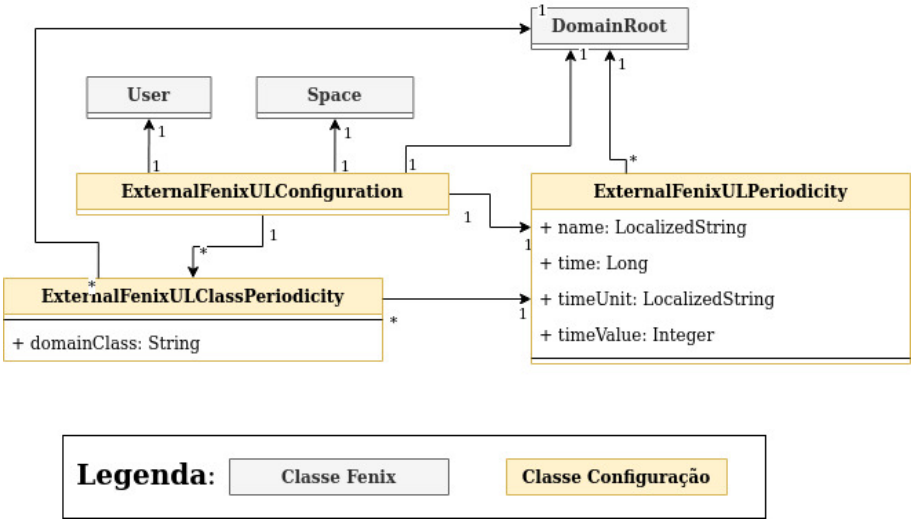

Figura 5.4: Diagrama de classes da parte de configuração dos cursos partilhados, onde em cinzento estão as classes pré-existentes no Fenix e a amarelo as novas classes de configuração

Na Figura 5.3 é apresentado um diagrama de sequência que exemplifica o funcionamento de um serviço *web*, utilizando o caso de uso de visualização do horários de um curso num semestre. O fluxo inicia com o pedido de um utilizador ao sistema Fenix, que durante o seu tratamento precisa de aceder aos turnos de uma disciplina de execução. No método responsável por fornecer estes dados é realizado uma invocação de um pedido ao serviço *web* da instância remota. No tratamento do serviço na instância remota é realizado o acesso aos turnos e às suas aulas. Esta informação é devolvida para a instância local que utiliza estes dados para apresentar o horário ao utilizador.

Na Figura 5.4 é apresentado o diagrama de classes da configuração de um curso partilhado. No sistema Fenix existe uma única instância do objeto de configuração, onde são associados o utilizador responsável pela atualização dos dados, o espaço onde são guardadas as salas externas, o valor por omissão da periodicidade de atualização das entidades externas e um conjunto de valores para periodicidades de classes específicas. A existência deste conjunto é justificada pelo facto de cada tipo de entidade ter uma frequência de alteração diferente, e desta forma podem ser atualizadas tendo em conta a sua frequência. Exemplificado com a disciplina de execução, os planos das aulas podem ser sujeitos a mais alterações ao longo do ano do que as referências bibliográficas. Desta forma, pode ser dada uma periodicidade maior à atualização dos planos.

5.3 Implementação

Devido à extensão do problema, nesta seção também apenas apresentamos os detalhes de implementação de um caso de uso, o caso de uso de visualização do horário de um

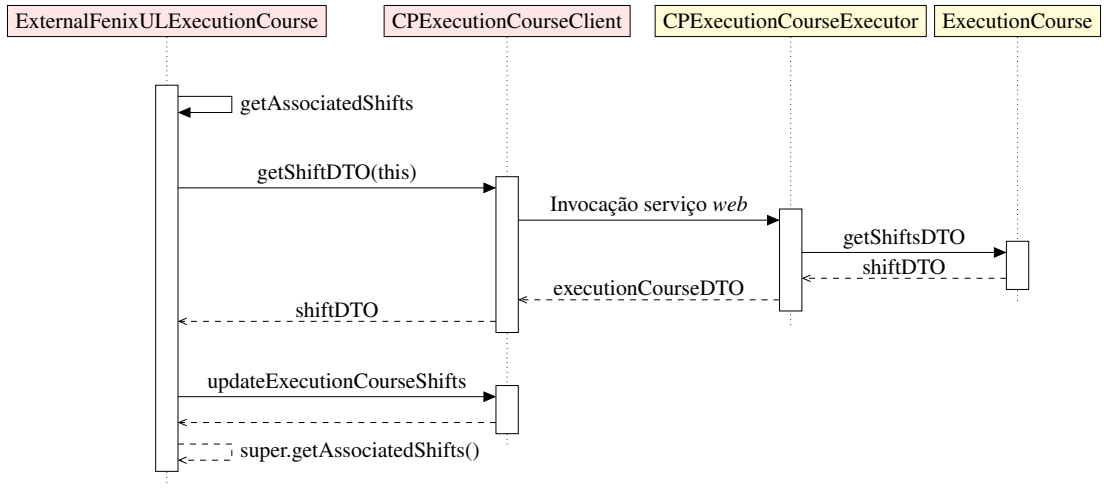

Figura 5.5: Diagrama de sequência da atualização dos turnos de uma *ExecutionCourse* remota. A vermelho estão representadas as classes locais e a amarelo as classes remotas.

curso num semestre (**CU13**).

A Figura 5.5 esquematiza o fluxo de execução entre as classes das instâncias local e remota do sistema Fenix. Esta execução é despoletada pela invocação do método *getAssociatedShifts()*.

Esta seção detalha a implementação do método de atualização dos turnos, da invocação do serviço *web* na instância local, do serviço *web* na instância remota e de um ecrã de exemplo. São também justificadas as opções de implementação.

5.3.1 Implementação do método de atualização dos turnos

A Listagem 5.1 exemplifica a implementação de um método utilizado para a atualização dos objetos, neste caso é utilizado o método *getAssociatedShifts*.

Inicialmente, o método *getAssociatedShifts* verifica a necessidade de atualização dos turnos (linhas 16 e 17). Caso seja necessária a atualização, são executas as linhas 29 a 31, que incluem a invocação remota do serviço *web*.

Para garantir que a invocação do serviço *web* e a atualização são realizadas por uma *thread* em exclusão mutua, foi utilizado um *ReentrantLock*, cuja lista de espera respeita a ordem de chegada das *threads*.

O *lock* é adquirido depois da verificação, ao contrário do que é normal, porque o resultado da verificação não é modificado pelas outras *threads*, a verificação é feita com campos definidos na DML e estes valores estão protegidos pela JVSTM. Esta inversão, do *lock* e verificação, permite que apenas as *threads* que precisam de atualizar os turnos fiquem bloqueadas no *lock*.

No entanto, apesar do *lock*, todas as *threads* cuja transação tem a versão desatualizada dos dados também invocariam o serviço *web* remoto tendo em vista a atualização dos mesmos. Para resolver este problema foi necessário definir uma variável fora da DML

de modo que o seu valor possa ser partilhado entre as *threads*. Esta variável, designada por *timeLastShiftDTO*, é também utilizada para verificar a necessidade de atualização dos turnos.

Para que as restantes *threads* passem a correr numa transação com uma versão atualizada dos dados é necessário forçar o seu recomeço. Isto é garantido com a modificação dos dados realizada na linha 36 que, não alterando o valor do campo, é suficiente para que no momento do *commit* seja verificada a versão da transação. Como não é a versão em que pode ser feito o *commit*, a JVSTM força a transação a recomeçar na versão correta.

O comportamento da JVSTM explicado anteriormente pode ocorrer também na primeira *thread* (a que vai tentar atualizar). Neste caso, a transação recomeçaria mas a variável *timeLastShiftDTO* ficaria com o valor atualizado, fazendo com que mais nenhuma *thread* atualizasse os dados corretamente antes do tempo de validade da cache expirar. Para corrigir este problema foi adicionado o *checkpoint* na linha 32, o qual tenta realizar o *commit*. No caso de não ter sucesso, é lançada uma exceção. No tratamento da exceção é apagado o valor da variável *timeLastShiftDTO*, permitindo que a próxima *thread*, ao adquirir no *lock*, possa fazer a atualização.

No caso da *thread* que faz a invocação remota não conseguir fazer a atualização, os dados são guardado na variável *shiftDTO* de modo a evitar a execução de invocações remotas adicionais. Desta forma, a *thread* que faz a atualização dos dados utiliza esta variável. Após ser feito o *checkpoint* com sucesso o valor da variável é apagado (linha 33). Este comportamento não está presente na Alternativa 0 avaliada na seção 6.2.1.

Na alternativa 2, avaliada na seção 6.2.1, foram adicionados os *commits* das linhas 37 e 46, de forma a forçar o recomeçar da transação, assim que falha a atualização dos turnos. No entanto, verificou-se que esta alternativa origina um maior número de recomeços para a mesma transação.

Este procedimento é exemplificado nos dois esquemas da Figura 5.6. No exemplo 1 são apresentadas duas *threads* t1 e t2 que entram no método com a versão 100 e 101 da JVSTM, respetivamente. Ambas têm os turnos desatualizados. A t1 entra no *lock* e a t2 fica bloqueada. A t1 atualiza o objeto, altera a variável *timeLastShiftDTO*, faz *checkpoint* da alteração e sai do *lock*. Neste momento a t2 entra no *lock*, falha a verificação executando assim a modificação da linha 36, quando tenta fazer o *commit*, a *JVSTM* verifica que não o pode fazer *commit* e reiniciando a transação com uma versão mais recente e com o objeto atualizado.

O exemplo 2 é parecido, mas neste caso é a t2 a entrar primeiro no *lock*, faz a atualização do objeto e quando faz o *commit*, a *JVSTM* verifica que não o pode fazer, forçando-a a reinicializar. Neste caso, para que não exista um bloqueio da transação, a variável *timeLastShiftDTO* é colocada a *null* no tratamento da excepção e a *thread* sai da zona de exclusão mutua. Neste momento, a *thread* t1 adquire o *lock* e executa-se o caso do exemplo 1.

```
1  public class ExternalFenixULExecutionCourse extends ExternalFenixULExecutionCourse_Base{
2      (...)
3      private DateTime timeLastShiftDTO;
4      private ReentrantLock lockShift = new ReentrantLock(true);
5      private ExecutionCourseDTO shiftDTO;
6      (...)
7      public ExecutionCourseDTO getShiftDTO() {
8          if(shiftDTO == null) {
9              shiftDTO = CPExecutionCourseClient.getShiftDTO(this);
10         }
11         return shiftDTO;
12     }
13     (...)
14     @Override
15     public Set<Shift> getAssociatedShifts() {
16         if (!getLockShift().isHeldByCurrentThread()
17                 && isToUpdate(getLastUpdateShifts(), Shift.class)) {
18             // Object cache expired and need update
19             getLockShift().lock();
20             try {
21                 Set<Shift> listShifts = super.getAssociatedShifts();
22                 ExternalFenixULExecutionCourse thisExecutionCourse = this;
23                 FenixFramework.getTransactionManager()
24                                 .withTransaction(new Callable<Object>() {
25                     @Override
26                     public Object call() throws Exception {
27                         if (isToUpdate(getTimeLastShiftDTO(), Shift.class)) {
28                            // Update shifts
29                            CPExecutionCourseClient
30                                    .updateExecutionCourseShifts
31                                         (thisExecutionCourse, listShifts, getShiftDTO());
32                             Transaction.checkpoint();
33                             setShiftDTO(null);
34                         } else {
35                             // Force re-execute transaction
36                             setLastUpdateShifts(getLastUpdateShifts());
37                             // Transaction.commit();
38                         }
39                         return null;
40                     }
41                 });
42             } catch (Exception e) {
43                 if(getTimeLastShiftDTO().equals(getLastUpdateShifts())) {
44                     setTimeLastShiftDTO(null);
45                 }
46                 // Transaction.commit();
47             } finally {
48                 getLockShift().unlock();
49             }
50         }
51         return super.getAssociatedShifts();
52     }
53     (...)
54
55 }
```

Listagem 5.1: Classe ExternalFenixULExecutionCourse com foco no método de atualização dos turnos.

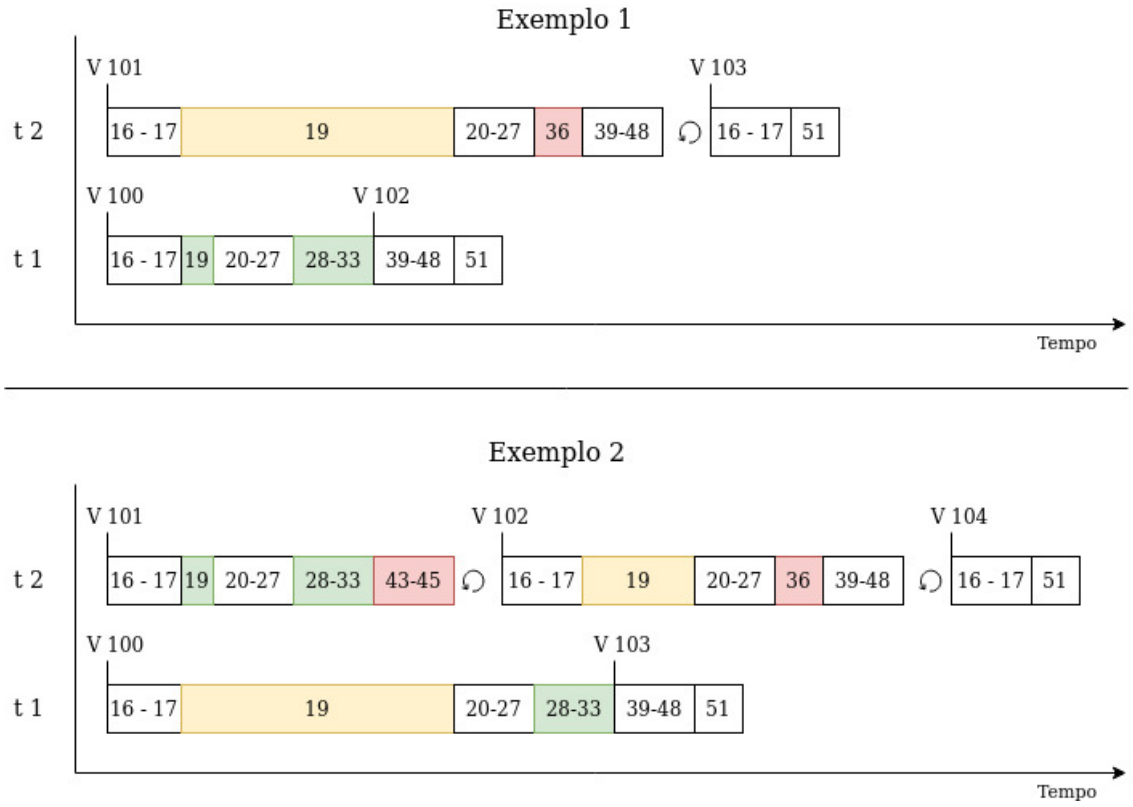

Figura 5.6: Exemplos de computação concorrente no método *getAssociatedShifts*, onde as caixas são blocos de código, com as linhas indicadas no mesmo. as versões da JVSTM aparecem no inicio do próximo bloco

5.3.2 Implementação da invocação do serviço *web* na instância local

Na implementação do serviço *web* na instância local foi criada a classe *CPExecutionCourseClient*, a qual é responsável pelos serviços relacionados com as disciplinas de execução. Nesta classe está definido o método *updateExecutionCourseShifts*, apresentado na Listagem 5.2.

Em alguns objetos de domínio é guardada a informação sobre os utilizadores que realizam as modificações. Neste caso não deve ser guardada essa informação, visto que não é o utilizador a fazer as alterações. De forma a garantir que isto não acontece, no início da atualização é feita a alteração de utilizador (linha 11), para o utilizador responsável pelas modificações feitas pelos serviços *web*. No final da atualização, volta a ser alterado o utilizador para o que estava autenticado inicialmente (linha 49).

Depois da primeira alteração do utilizador, é feita a modificação dos dados existentes para os dados que estão no *Domain Tranfer Object* (DTO), atualizando os turnos. No entanto, pode acontecer que existam métodos de *set* que apenas mantém o valor e a *framework* Fenix considera-os também escritas à base de dados. Desta forma, para melhorar o desempenho da atualização, reduzindo o número de escritas feitas à base de dados,

antes de cada alteração é feita uma verificação da necessidade de realizar a modificação (exemplo na linha 19).

Nesta classe é também implementado o método *getShiftDTO*, que realiza a invocação remota. Antes de fazer a invocação remota é criado um *RequestPropertiesDTO*, utilizado para enviar as informações necessárias para o processamento do serviço na instância remota. De seguida é realizada invocação remota que devolve um objeto do tipo *ExecutionCourseDTO*. No caso de não existir nenhum erro no processamento do serviço, o valor da variável *timeLastShiftDTO* é atualizado.

```
1  public class CPExecutionCourseClient {
2
3      (...)
4
5      public static void updateExecutionCourseShifts
6                          (ExternalFenixULExecutionCourse executionCourse,
7                          Set<Shift> listShifts, ExecutionCourseDTO executionCourseDTO) {
8          if (executionCourseDTO == null) {
9              return;
10         }
11         User loggedUser = setDefaultUser();
12         // Update shifts
13         List<ShiftDTO> listShiftsDTO = executionCourseDTO.getShifts();
14         listShifts.forEach(shift -> {
15             ShiftDTO shiftDTO = getShiftDTO(listShiftsDTO, shift);
16             if (shiftDTO == null) {
17                 shift.delete();
18             } else {
19                 if (updateShiftIsNeeded(shift, shiftDTO)) {
20                     editShift(executionCourse, shift, shiftDTO);
21                 }
22                 // Update Lessons
23                 List<LessonDTO> listLessonDTO = shiftDTO.getLessons();
24                 shift.getAssociatedLessonsSet().forEach(lesson -> {
25                     LessonDTO lessonDTO = getLessonDTO(listLessonDTO, lesson);
26                     if (lessonDTO == null) {
27                         lesson.delete();
28                     } else {
29                         if (updateLessonIsNeeded(lesson, lessonDTO)) {
30                             editLesson(lesson, lessonDTO);
31                         }
32                         listLessonDTO.remove(lessonDTO);
33                     }
34                 });
35                 // create new Lessons
36                 listLessonDTO.forEach(lessonDTO -> {
37                     createLesson(shift, lessonDTO);
38                 });
39                 listShiftsDTO.remove(shiftDTO);
40             }
41         });
42         // Create new shifts
43         listShiftsDTO.forEach(shiftDTO -> {
44             Shift shift = createShift(executionCourse, shiftDTO);
45             shiftDTO.getLessons().forEach(lessonDTO -> {
46                 createLesson(shift, lessonDTO);
47             });
48         });
49         setAuthenticatedUser(loggedUser);
50         executionCourse.setLastUpdateShifts(executionCourse.getTimeLastShiftDTO());
51     }
52
53     (...)
54
```

```
55      public static ExecutionCourseDTO
56                          getShiftDTO(ExternalFenixULExecutionCourse executionCourse) {
57          RequestPropertiesDTO requestProperties = new RequestPropertiesDTO();
58          requestProperties.addProperty("ExecutionYear", executionCourse
59                                                  .getExecutionYear().getQualifiedName());
60          requestProperties.addProperty("TypeInfo", "shift");
61
62          ExecutionCourseDTO executionCourseDTO =
63                  CPExecutionCourseClient.getExecutionCourseInfo(
64                              CPClientUtils.getExternalInstituitionURL(executionCourse),
65                              CPClientUtils.getDegreeCode(executionCourse),
66                              CPClientUtils.getExternalCourseCode(executionCourse),
67                              requestProperties);
68          if (executionCourseDTO.getErrorCode() != null) {
69              return null;
70          }
71          executionCourse.setTimeLastShiftDTO(DateTime.now());
72          return executionCourseDTO;
73      }
74
75      (...)
76
77  }
```

Listagem 5.2: Cliente responsável pelo pedido dos serviços das disciplinas de execução.

5.3.3 Implementação do serviço *web* na instância remota

Na instância Fenix remota existe a classe *CPExecutionCourseExecutor* responsável pelo serviço *web* relacionado com disciplinas de execução. Na Listagem 5.3 é apresentado um excerto desta classe.

O método *execute* deve ser definido para implementar o tratamento deste serviço específico. Para reduzir o número de serviços existentes e facilitar a sua gestão, os vários pedidos relacionados com a disciplina de execução foram agrupados num único serviço. Para diferenciar o pedido existe o parâmetro *requestTypeInfo*.

O tratamento do pedido, neste exemplo do tipo *SHIFT*, retorna uma lista com os dados referentes aos turnos e respetivas aulas, que é adicionada ao *executionCourseDTO*, resultado do método. Na classe *RestExecutor*, o *executionCourseDTO* é mapeado para *JSON* e enviado na resposta do serviço *web*.

```
1   public class CPExecutionCourseExecutor implements RestExecutor<ExecutionCourseDTO> {
2       (...)
3       @Override
4       public ExecutionCourseDTO execute(RestRequest request) {
5           ExecutionCourseDTO executionCourseDTO = new ExecutionCourseDTO();
6           try {
7               String requestCourse = request.getPathParameter(COURSE);
8               String requestdegree = request.getPathParameter(DEGREE);
9               RequestPropertiesDTO properties = CPExecutorUtils.getProperties(request);
10              if (properties == null) {
11                  // Missing  requestPropertiesDTO from webservice
12                  executionCourseDTO.setErrorCode
13                                          (CPExecutorUtils.CODE_ERROR_MISSING_PARAMS);
14                  return executionCourseDTO;
15              }
16
17              String requestTypeInfo = properties.getProperty(TYPE_INFO);
```

```
18              ExecutionCourse executionCourse =
19                      getExecutionCourse(requestCourse, requestdegree, properties);
20
21              switch (requestTypeInfo) {
22              case BIBLIOGRAPHIC_REFERENCE:  // Get BIBLIOGRAPHIC_REFERENCE Info
23                  executionCourseDTO.setBibliographicReferences
24                                  (getBibliographicReferencesDTO(executionCourse));
25                  break;
26              case LESSON_PLANNING:          // Get LESSON_PLANNING Info
27                  executionCourseDTO.setLessonPlannings
28                                      (getLessonPlanningsDTO(executionCourse));
29                  break;
30              case SHIFT:                    // Get SHIFT Info
31                  executionCourseDTO.setShifts(getShiftsDTO(executionCourse));
32                  break;
33              default:
34                  break;
35              }
36          } catch (Exception e) {
37              executionCourseDTO.setErrorCode(e.getMessage());
38              return executionCourseDTO;
39          }
40          return executionCourseDTO;
41      }
42      (...)
43  }
```

Listagem 5.3: Executor responsável por tratar os pedidos relacionados com a disciplina de execução.

5.3.4 Implementação de um ecrã

Nesta seção é apresentado um exemplo da criação de ecrãs utilizando a PSL da *framework OMNIS*.

Para este exemplo relembramos parte do exemplo utilizado no Capítulo 2, relativo à linguagem de modelação da apresentação. Na Listagem 5.4 são descritos dois ecrãs pertencentes ao mesmo fluxo, o *searchExternalFenixULPeriodicity* e o *createExternalFenixULPeriodicity*. O segundo é acessível através do evento de criação do primeiro e é apresentado como *dialog*.

O ecrã *createExternalFenixULPeriodicity* é do tipo *createScreen* e apresenta os campos *name, timeUnit e timeValue* da classe *ExternalFenixULPeriodicity*. Este ecrã tem ainda dois eventos, o de criar e o de cancelar.

```
1  (...)
2  flow manageExternalFenixULPeriodicity channel(WebJava) {
3      searchScreen searchExternalFenixULPeriodicity [ExternalFenixULPeriodicity]
4                                          (fields name time){
5          dialog createEvent -> createExternalFenixULPeriodicity
6          (...)
7      }
8      createScreen createExternalFenixULPeriodicity [ExternalFenixULPeriodicity]
9                                          (fields name timeUnit timeValue){
10         createEvent
11         cancelEvent
12     }
13     (...)
14 }
```

```
15 (...)
```

Listagem 5.4: Excerto da PSL do projeto com ênfase no ecrã de criação de periodicidade.

Durante a compilação do projeto é criada a classe *CreateExternalFenixULPeriodicity*, exemplificada na Listagem 5.5, que apresenta algumas alterações ao ecrã *base*. Os campos do ecrã são modificados no método *getSchema*. Neste caso são definidos como obrigatórios e no campo *timeUnit* é adicionada a lista de unidades de tempo a apresentar.

Os métodos que começam com *process* seguido do tipo de evento são utilizados para modificar o comportamento dos eventos. No *processCreateEvent* é executada uma transação para a criar um objeto do tipo *ExternalFenixULPeriodicity*.

```
1  public class CreateExternalFenixULPeriodicity extends
2                              CreateExternalFenixULPeriodicity_Base {
3      @Override
4      protected Schema getSchema() {
5          Schema schema = super.getSchema();
6          schema.getSchemaSlot("name").caption(i18n("externalFenixULPeriodicity.name"))
7                      .required();
8          schema.getSchemaSlot("timeUnit")
9                      .caption(i18n("externalFenixULPeriodicity.timeUnit")).required()
10                     .addOptions(ExternalFenixULPeriodicity.getTimeUnits());
11         schema.getSchemaSlot("timeValue")
12                     .caption(i18n("externalFenixULPeriodicity.timeValue")).required();
13         return schema;
14     }
15     @Override
16     protected void processCreateEvent(SaveEvent event)  {
17         executeInTransactionalContext((Runnable) () -> {
18             final ExternalFenixULPeriodicity proxy =
19                         getObjectCreator().getProxy(ExternalFenixULPeriodicity.class);
20             ExternalFenixULPeriodicity.create(
21                 getRealObject(proxy.getName()),
22                 proxy.getTimeValue(),getRealObject(proxy.getTimeUnit()));
23         })
24         this.getContainingModalWindow().close();
25     }
26     @Override
27     protected void processCancelEvent(CancelEvent event)  {
28         this.getContainingModalWindow().close();
29         super.processCancelEvent(event);
30     }
31 }
```

Listagem 5.5: Classe gerada pela PSL para o ecrã de criação de tipos de periodicidade.

O ecrã apresentado na Figura 5.7 é gerado com o código da Listagem 5.5. Na imagem são visíveis as alterações aos campos: todos os campos são obrigatórios e o campo *Unidade de tempo* tem uma *combobox* com a lista de unidades de tempo.

No final da Figura 5.7 são apresentados os botões responsáveis pelos respetivos eventos. Quando utilizado, o botão de Criar vai executar o método *processCreateEvent* e o botão Cancelar o método *processCancelEvent* presentes na Listagem 5.5.

Figura 5.7: Exemplo do ecrã gerado a partir da classe *CreateExternalFenixULPeriodicity*

5.4 Sumário

Neste capítulo são apresentadas as alternativas de concretização do projeto ao nível da arquitetura e do desenho. Destas foram tomadas decisões para a implementação de uma prova de conceito, exemplificada na seção de implementação. A prova de conceito, é utilizada no capítulo seguinte para a validação do projeto.

Capítulo 6

Validação

A validação da proposta apresentada neste trabalho é focada no requisito de desempenho. Os testes efetuados na instância local avaliam o desempenho de três alternativas de implementação. Adicionalmente o desempenho da alternativa de implementação escolhida é estudado com testes de carga fazendo variar o número de utilizadores e o prazo de validade de cache. Os testes efetuados na instância remota avaliam o desempenho da alternativa escolhida é estudado com testes de carga fazendo variar o número de pedidos realizados.

6.1 Introdução aos testes

Os testes foram realizados através da ferramenta Apache-Jmeter, a qual permite realizar testes de desempenho simulando utilizadores a executar uma ou mais operações num *website*.

Os testes decorreram utilizando três máquinas, que estão descritas na Tabela 6.1. A máquina local é responsável por uma réplica da instância Fenix da Faculdade de Farmácia e é onde correm os testes na aplicação JMeter. A máquina remota é uma máquina virtual responsável por uma réplica da instância Fenix do Instituto de Geografia e Ordenamento do Território. Por último, temos a máquina onde está a base de dados das duas instâncias, configurada com um número máximo de 151 ligações em simultâneo.

O tempo de execução de cada teste é obtido através da média dos tempos obtidos pela execução de 5 repetição.

Máquina	**Tipo de Máquina**	**CPUs**		**RAM**	**BD ligações**
		Nº cores	**Frequência**		
Instância local	Desktop	6	3.2 Ghz	32 Gb	
Instância remota	VM	2	2.2 Ghz	16 Gb	
Base de dados	VM	1	2.2 Ghz	8 Gb	151 ligações

Tabela 6.1: Caracterização das máquinas utilizadas para os testes

6.2 Testes de desempenho na instância local

Nos testes de carga da instância local foi utilizado o caso de uso de visualizar o horário de um semestre do curso (**CU13**) e foram realizados testes para avaliar o comportamento da instância tendo em conta as seguintes vertentes de avaliação: alternativas de implementação do código, o número de utilizadores e o prazo de validade da cache. O semestre do curso utilizado tem as cinco UCs externas, assim sendo os valores encontrados refletem o tempo utilizado para gerar as cinco atualizações.

Em cada um dos casos, os testes foram repetidos cinco vezes e com os resultados obtidos foram calculados os extremos e os quartis, bem como a percentagem respostas obtidas tendo em conta os intervalos pré-definidos seguintes:

- **(0-50ms)** - Neste intervalo assume-se que para dar resposta ao pedido foi utilizada a cache.
- **(mais de 150 ms)** - Neste intervalo assume-se que foram feitas as atualizações da cache por umas *threads* e as restantes ficaram à espera que esta estivesse concluida.
- **50-150** - Neste intervalo estão os tempos de resposta que tanto podem ser acessos à cache mais lentos ou respostas que esperam pelas atualizações menos tempo.

Os valores referentes ao tempo estão em milissegundos e o número de utilizadores é por minuto. Antes da realização de cada um dos testes foram carregados para memória os dados a apresentar.

6.2.1 Testes de desempenho para as alternativas de implementação

Neste teste é feita a comparação entre três alternativas de implementação, apresentadas de seguida:

- **Alternativa 0** - Nesta alternativa os DTO não são guardados. Desta forma, todos os utilizadores fazem a invocação do serviço *web* remoto quando tentam atualizar a cache.
- **Alternativa 1** - Nesta alternativa é guardado o DTO. Desta forma, apenas uma *thread* faz a invocação do serviço *web* remoto.
- **Alternativa 2** - Este alternativa refere-se à realização dos *commits* após a verificação que o utilizador não está a utilizar a versão que permite realizar os mesmos.

Estas alternativas são detalhadas na seção 5.3.1.

Estes testes tem como objetivo a escolha da melhor opção para a implementação do método *getAssociatedShifts*, referenciado na seção de implementação do capítulo anterior (seção 5.3.1). Para isto, os testes foram feitos com 1000 utilizadores por minuto e o prazo de validade da cache foi de 10s.

Na Figura 6.1 são apresentados graficamente os resultados dos testes que estão presentes na Tabela 6.2. No gráfico não é possível ver uma grande diferença que permita fazer uma escolha, apesar do valor máximo da Alternativa 1 ser quase metade das outras duas.

No entanto, na Figura 6.2 é representado o gráfico construido a partir da Tabela 6.3 Neste é possível verificar que a alternativa 1 tem uma maior percentagem de utilizadores, com tempos de resposta no intervalo 0-50 ms (acesso à cache) e apenas 1,84% do utilizadores com tempos de resposta no intervalo de mais de 150 ms (atualizam os dados). Tendo em conta estes valores optou-se por implementar esta alternativa.

Alternativas	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
Alternativa 0	34	37	38	41	690
Alternativa 1	34	37	38	41	360
Alternativa 2	33	37	39	42	680

Tabela 6.2: Resultados dos testes de desempenho para as alternativas de implementação, com o cálculo do mínimo, máximo e quartis. Os valores estão em milissegundos.

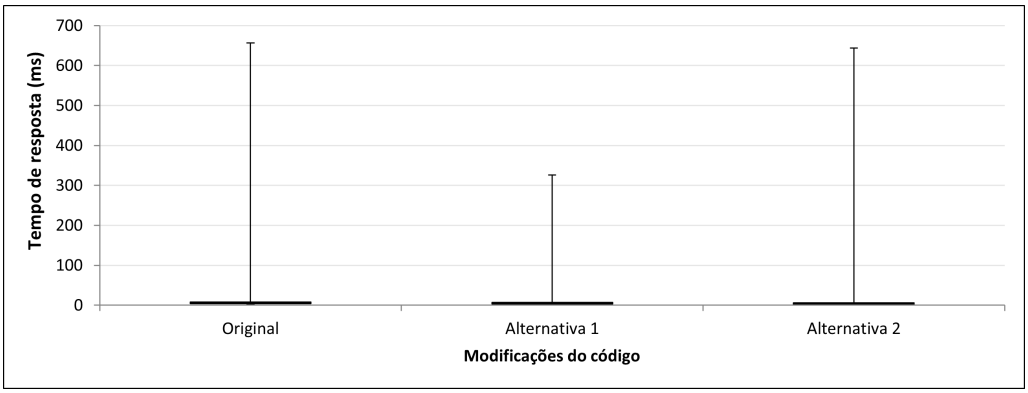

Figura 6.1: Gráfico dos testes de desempenho para as alternativas de implementação, com o cálculo do mínimo, máximo e quartis.

Alternativas	0-50 ms	50-150 ms	150+ ms
Alternativa 0	91,42%	6,22%	2,36%
Alternativa 1	94,12%	4,04%	1,84%
Alternativa 2	87,62%	10,28%	2,10%

Tabela 6.3: Resultados dos testes de desempenho para as alternativas de implementação, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-50 ms, 50-150 ms e mais de 150 ms.

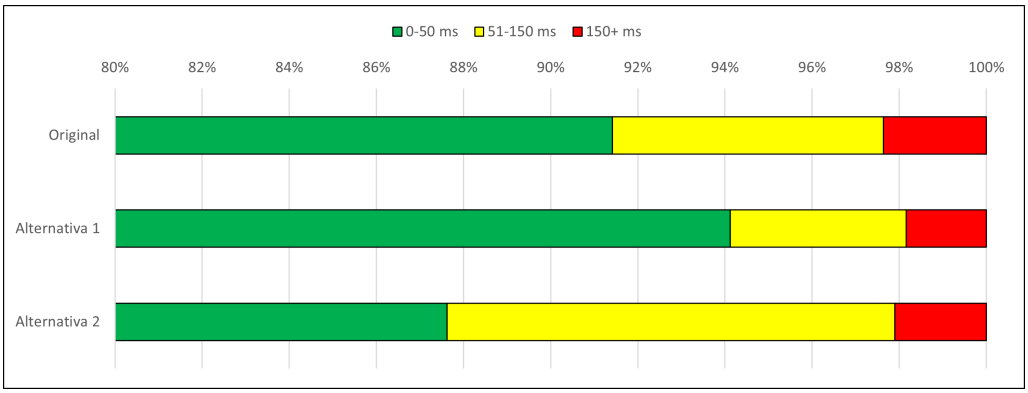

Figura 6.2: Gráfico dos testes de desempenho para as alternativas de implementação, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-50 ms, 50-150 ms e mais de 150 ms.

6.2.2 Testes de desempenho em função do número de utilizadores

Estes testes têm como objetivo avaliar o comportamento da implementação escolhida, face ao aumento do número de utilizadores. Nestes testes, o prazo de validade da cache utilizado foi de 10s.

A Figura 6.3 representa graficamente os valores da Tabela 6.4, referentes aos extremos e quartis encontrados nos testes de cada grupo de utilizadores. Observa-se que o valor máximo não aumenta com o aumento do números de utilizadores. Observa-se ainda uma diminuição da distância entre o 1º e 3º quartil à medida que o número de utilizadores aumenta, facto justificado pelo aumento do número de pedidos que acedem à cache.

Na Figura 6.4 é representado o gráfico construído a partir da Tabela 6.5. Nos valores apresentados no gráfico destaca-se a diminuição da percentagem de utilizadores que fazem ou esperam pela atualização dos dados à medida que o número de utilizadores cresce; esta percentagem desce de 50% com 10 utilizadores para cerca de 2% com 1000.

Este teste permite concluir que a solução não tem um impacto demasiado grande no normal funcionamento do Fenix, visto que os valores dos piores casos estão entre os 330 e os 450 milissegundos. A maioria dos utilizadores acede à cache, tendo assim um tempo de resposta semelhante ao do acesso local, à exceção do caso em que o número de utilizadores é inferior a 10, inclusive, onde o número de utilizadores que acede à cache é inferior ou igual aos que fazem a atualização dos dados.

Utilizadores	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
10/min	41	48	166,5	311	386
25/min	39	47	55	85	397
50/min	39	49	58	75	436
100/min	37	44	50	64	386
250/min	36	48	55	65	363
500/min	35	45	49	56	331
1000/min	34	37	38	41	360

Tabela 6.4: Resultados dos testes de desempenho em função do número de utilizadores por minuto, com o cálculo do mínimo, máximo e quartis. Os valores estão em milissegundos.

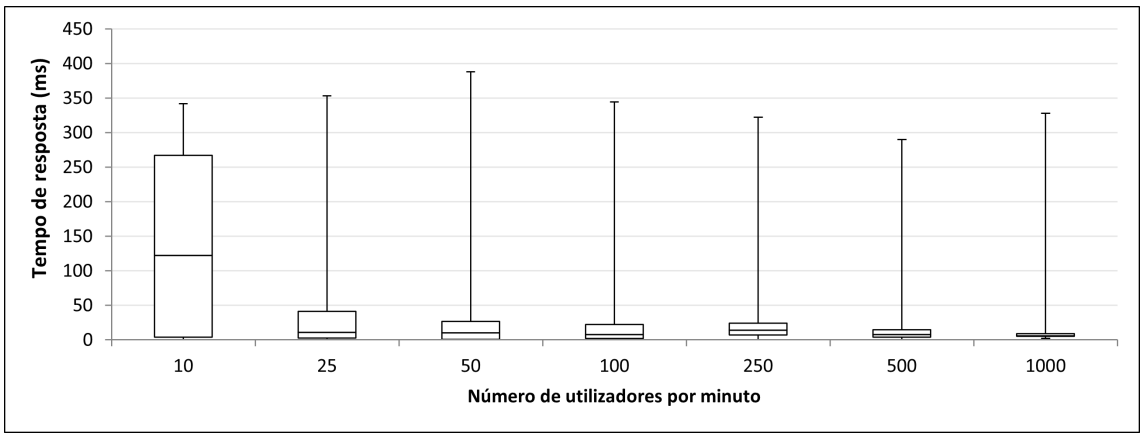

Figura 6.3: Gráfico dos testes de desempenho em função do número de utilizadores por minuto, com o cálculo do mínimo, máximo e quartis.

Utilizadores	0-50 ms	50-150 ms	150+ ms
10/min	36,00%	14,00%	50,00%
25/min	36,80%	43,20%	20,00%
50/min	32,00%	56,00%	12,00%
100/min	52,60%	41,40%	6,00%
250/min	35,44%	62,00%	2,56%
500/min	54,76%	43,32%	1,92%
1000/min	94,12%	4,04%	1,84%

Tabela 6.5: Resultados dos testes de desempenho em função do número de utilizadores por minuto, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-50 ms, 50-150 ms e mais de 150 ms.

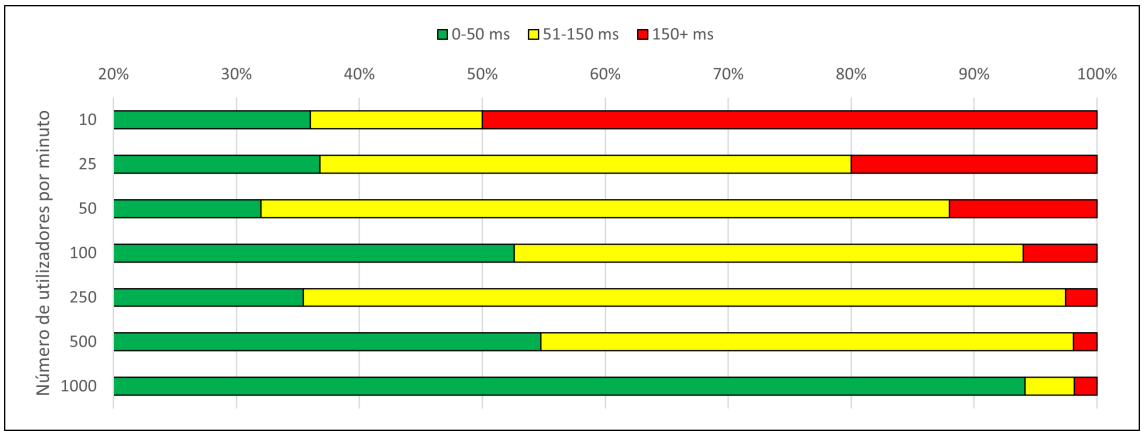

Figura 6.4: Gráfico dos testes de desempenho em função do número de utilizadores por minuto, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-50 ms, 50-150 ms e mais de 150 ms.

6.2.3 Testes de desempenho em função do prazo de validade da cache

Estes testes têm como objetivo avaliar o impacto do prazo de validade da cache nos tempos de acesso obtidos. Nestes testes, foram utilizados 1000 utilizadores por minuto.

A Figura 6.5 são apresentados graficamente os resultados dos testes que estão presentes na Tabela 6.6. No gráfico observa-se a diminuição da diferença entre o 1º e 3º quartil, isto significa que 25% dos utilizadores tiveram um tempo de resposta maior com 1s de cache em comparação com as restantes. No entanto, na Tabela 6.7 e no respetivo gráfico, Figura 6.6, esta diferença está mais acentuada. Enquanto que com 1s, 71% dos utilizadores acedem à cache, com 30s este número sobe para 95%. Pelo contrário, a percentagem de utilizadores que fazem ou esperam pela atualização dos dados diminuem de cerca de 2,7% para cerca de 0,5%.

Estes testes permitem verificar que o impacto do prazo de validade da cache é maior quando o valor de cache está entre 1 e 10 segundos. A partir dos 10 segundos, o aumento do prazo de validade da cache não tem tanto impacto. Desta forma é possível concluir que com 10s de cache, é possível ter os dados relativamente atualizados e sem grande impacto no desempenho do sistema Fenix.

Cache	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
1s	34	38	40	57	425
10s	34	37	38	41	360
30s	34	38	40	42	395

Tabela 6.6: Resultados dos testes de desempenho em função do prazo de validade da cache, com o cálculo do mínimo, máximo e quartis. Os valores estão em milissegundos.

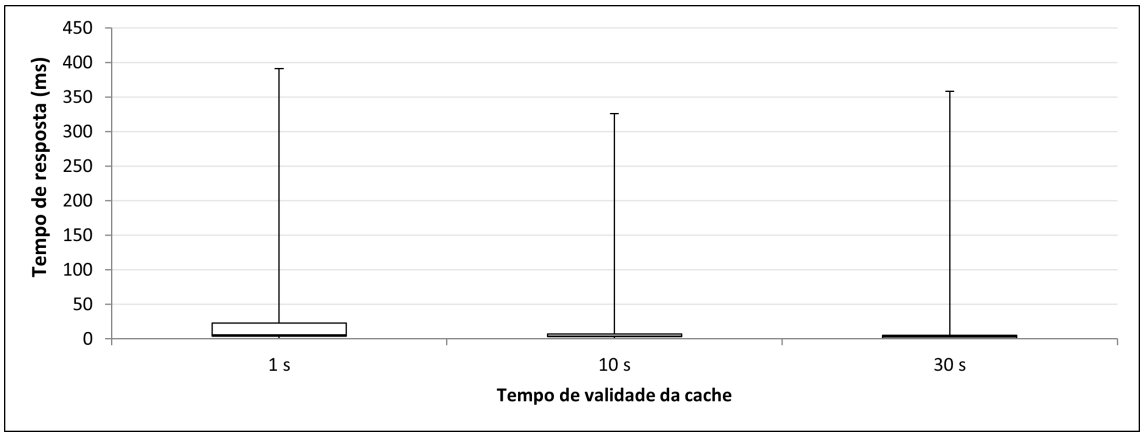

Figura 6.5: Gráfico dos testes de desempenho em função do prazo de validade da cache, com o cálculo do mínimo, máximo e quartis.

Cache	0-50 ms	50-150 ms	150+ ms
1s	71,76%	25,50%	2,74%
10s	94,12%	4,04%	1,84%
30s	95,22%	4,24%	0,54%

Tabela 6.7: Resultados dos testes de desempenho em função do prazo de validade da cache, com a percentagem de utilizadore com tempos de resposta nos intervalos de 0-50 ms, 50-150 ms e mais de 150 ms.

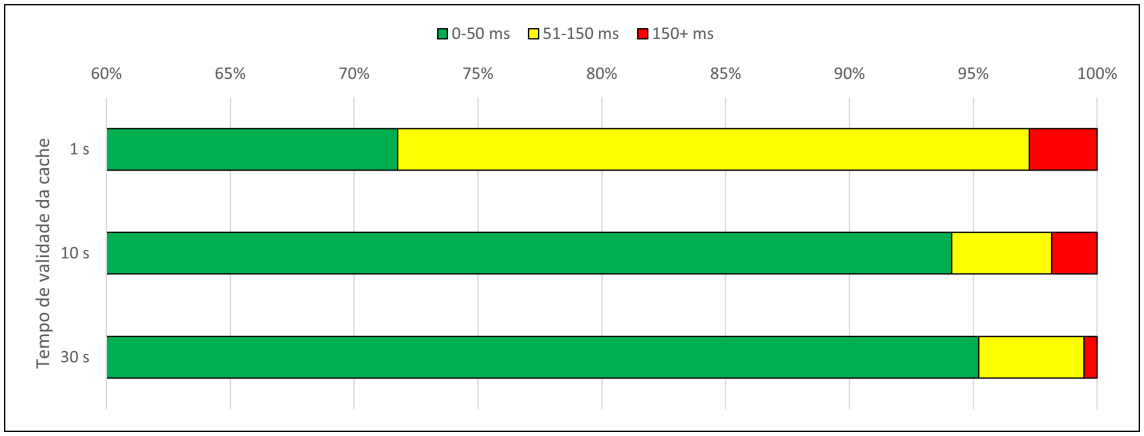

Figura 6.6: Gráfico dos testes de desempenho em função do prazo de validade da cache, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-50 ms, 50-150 ms e mais de 150 ms.

6.3 Testes de desempenho na instância remota

Estes testes têm com objetivo avaliar o impacto dos pedidos de serviços *web* no desempenho da instância remota. Nestes testes foram feitos vários pedidos do mesmo serviço num minuto.

Na Figura 6.7 é apresentado o gráfico com os resultados dos testes que estão presentes na Tabela 6.8, no qual se observa um pequeno crescimento dos valores máximos, contínuo desde os 10 pedidos com 27 ms até aos 1000 pedidos com 89 ms. Nesta análise são excluídos os valores de 25 e 500 pedidos que são casos isolados.

Na Figura 6.8 são apresentados os valores da Tabela 6.9. Para a grande maioria dos pedidos, o tempo de resposta é inferior a 25 ms. No pior caso, 25 pedidos por min, apenas 0,8% dos resultados têm um tempo de resposta superior a 50 ms. Desta forma, é possível concluir que esta solução não apresenta impacto no desempenho na instância remota do sistema Fenix.

Pedidos	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
10/min	10	13	13	16	27
25/min	8	11	12	13	143
50/min	7	9	10	11	27
100/min	7	9	10	11	55
250/min	7	8	9	10	52
500/min	7	8	8	9	212
1000/min	6	7	8	9	89

Tabela 6.8: Resultados dos testes de desempenho de invocações feitas diretamente à instância remota, com o cálculo do mínimo, máximo e quartis. Os valores estão em milissegundos.

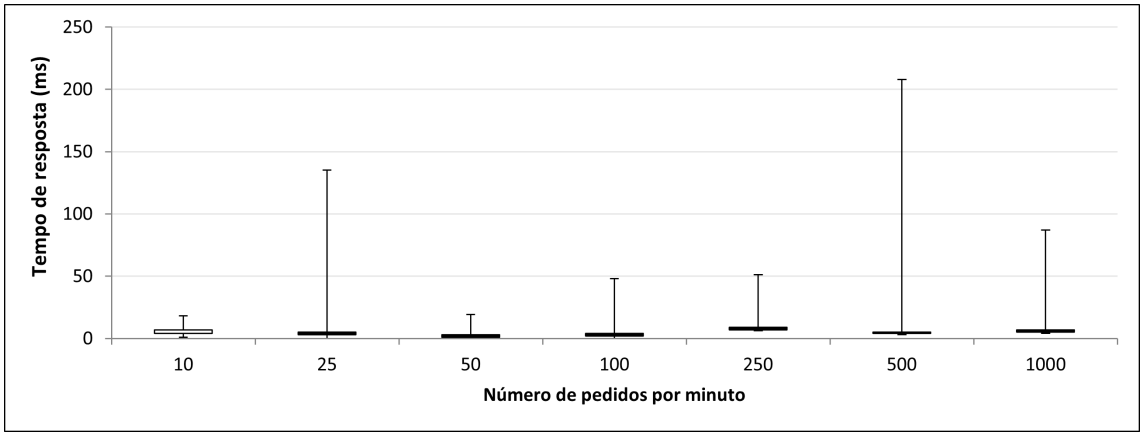

Figura 6.7: Gráfico dos testes de desempenho de invocações feitas diretamente à instância remota, com o cálculo do mínimo, máximo e quartis.

Utilizadores	0-50 ms	25-50 ms	50+ ms
10/min	92,00%	8,00%	0,00%
25/min	99,20%	0,00%	0,80%
50/min	99,60%	0,40%	0,00%
100/min	97,60%	2,20%	0,20%
250/min	98,40%	1,52%	0,08%
500/min	99,28%	0,68%	0,04%
1000/min	99,60%	0,32%	0,08%

Tabela 6.9: Resultados dos testes de desempenho de invocações feitas diretamente à instância remota, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-25 ms, 25-50 ms e mais de 50 ms.

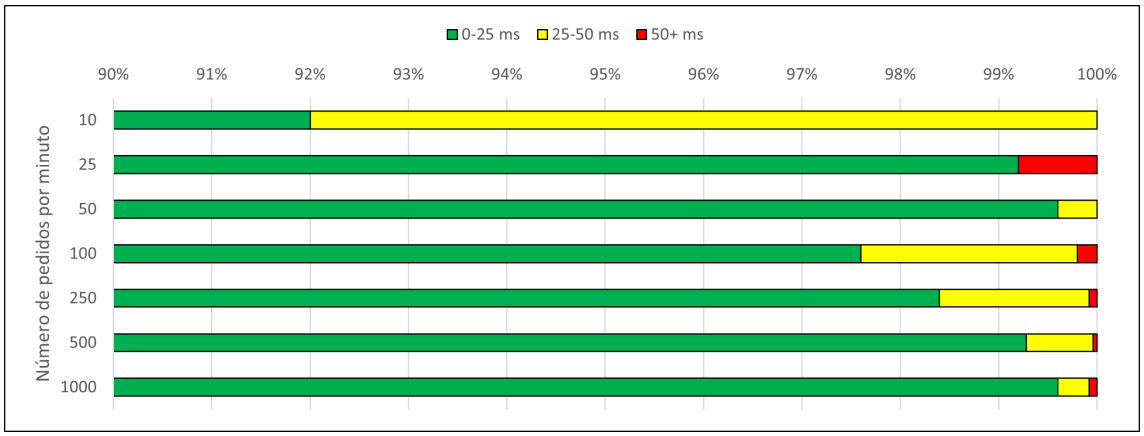

Figura 6.8: Gráfico dos testes de desempenho de invocações feitas diretamente à instância remota, com a percentagem de utilizadores com tempos de resposta nos intervalos de 0-25 ms, 25-50 ms e mais de 50 ms.

6.4 Sumário

Neste capítulo são descritos os testes realizados às instâncias local e remota do Fenix e os respetivos resultados, permitindo concluir que a proposta apresentada no capítulo 5 para a implementação no Fenix da funcionalidade de suporte à gestão de cursos partilhados é viável.

Capítulo 7

Conclusões e trabalho futuro

Atualmente, o Sistema Integrado de Gestão Académica da ULisboa, designado por Fenix, não suporta a gestão de cursos partilhados. As ações que envolvem os cursos partilhados são feitas manualmente pelos Serviços Académicos das escolas. Por esta razão, surgiu a necessidade da realização deste trabalho, que tem como objetivo a análise de alternativas para a concretização da solução desta problemática.

Antes de ser realizada a análise das alternativas, foi realizado o levantamento de requisitos em colaboração com o Departamento de Informática dos Serviços Centrais e com os serviços académicos de algumas escolas, nomeadamente a Faculdade de Letras da ULisboa e o Instituto de Geografia e Ordenamento do Território. Este levantamento de requisitos permitiu conhecer o funcionamento dos cursos partilhadas na ULisboa e definir os requisitos e os principais casos de uso.

Conhecidos os requisitos e os principais casos de uso, foram estudadas várias alternativas para a arquitetura e desenho do sistema. Optou-se por uma arquitetura de sistema *peer-to-peer* com o estilo arquitetural baseado em objetos e serviços, resolvendo os problemas de coerência, originados pelas escritas sobre objetos remotos, ao nível da camada aplicacional, onde as transações de escrita devem conter procedimentos para desfazer alterações remotas. Ao nível do desenho, optou-se por se utilizar o mecanismo de herança do Java para distinguir entidades locais das entidades remotas.

Esta proposta foi implementada, como prova de conceito, de forma a validar a possibilidade de ser utilizada para dar resposta ao problema dos cursos partilhados no Fenix. Durante esta implementação foram estudadas algumas opções com o objetivo de evitar que várias *threads* fizessem a atualização da cache do mesmo objeto ao mesmo tempo e assegurar a interligação deste objetivo com os mecanismos transacionais do sistema Fenix.

Os testes realizados na validação da prova de conceito permitem concluir que a proposta apresentada pode ser utilizada para a concretização da solução de gestão dos cursos partilhados no Fenix. Isto é justificado pelo facto da proposta não ter um grande impacto no normal desempenho da instância local, nem no desempenho da instância remota do

Fenix.

Tendo por base os resultados deste trabalho, pretende-se adicionar as funcionalidades relativas à gestão de cursos partilhados ao sistema Fenix de ULisboa. A entrada em produção destas funcionalidades carece ainda da realização de testes de aceitação e usabilidade, os quais irão permitir também aferir a completude da solução tendo em conta a diversidade e as especificações das várias escolas da ULisboa.

Bibliografia

[1] Joao Cachopo. *Development of rich domain models with atomic actions.* PhD thesis, Instituto Superior Técnico, 2007.

[2] Scott Chacon. Git. `https://git-scm.com/about`. (Acedido em 29/11/2019).

[3] Sérgio Fernandes. *Strongly Consistent Transactions for Enterprise Applications.* PhD thesis, Instituto Superior Técnico, 2014.

[4] The Apache Software Foundation. Apache maven project. `https://maven.apache.org/what-is-maven.html`. (Acedido em 29/11/2019).

[5] The Apache Software Foundation. Tomcat 7. `http://tomcat.apache.org/`. (Acedido em 29/11/2019).

[6] Oracle. Java 8. `https://java.com/en/download/faq/java8.xml`. (Acedido em 29/11/2019).

[7] Oracle. Mysql. `https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html`. (Acedido em 29/11/2019).

[8] Fundação para a ciência e a tecnologia. Fénix framework. `http://fenix-framework.github.io/`. (Acedido em 02/12/2019).

[9] António Rito Silva, Artur Ventura, Carlos Ribeiro, Fernando Mira da Silva, João Cachopo, and Luís Cruz e Susana Fernandes. *The FenixEdu Project: an Open-Source Academic Information Platform*. Instituto Superior Técnico, 2011.

[10] Instituto Superior Técnico. Fenixedu. `https://confluence.fenixedu.org/display/FENIXEDU/Welcome`. (Acedido em 02/12/2019).

[11] Vaddin. Vaddin docs.
`https://vaadin.com/docs/index.html`. (Acedido em 20/09/2020).

[12] Maarten van Steen e Andrew S. Tanenbaum. *Distributed Systems*. Maarten van Steen, 3 edition, 2018.

Acrónimos

ULisboa Universidade de Lisboa

Fenix Sistema FenixEdu

IST Instituto Superior Técnico

JVM *Java Virtual Machine*

RDBMS *Relational Database Management System*

JSP *Java Server Pages*

DML *Domain Modeling Language*

DSL *Domain Specific Language*

PSL *Presentation Specific Language*

MVC *Model-View-Controller*

CRUD *Create, Read, Update and Delete*

JVSTM *Java Versioned Software Transactional Memory*

SA Serviços Académicos

UC Unidade Curricular

RAIDES Registo de Alunos Inscritos e Diplomados do Ensino Superior

XML *EXtensible Markup Language*

PRIES Plataforma de Recolha de Informação do Ensino Superior

DI SC Departamento de Informática dos Serviços Centrais

SIGA Sistema Integrado de Gestão Académica

SAS Serviços de Ação Social

DSS Diagramas de Sequência do Sistema

COS Contratos das Operações do Sistema

ECTS European Credit Transfer System

DGES Direção-Geral de Ensino Superior

TCP *Transmission Control Protocol*

SOAP *Simple Object Access Protocol*

REST *REpresentational State Transfer*

DTO *Domain Tranfer Object*

Apêndice A

Modelo de classes de domínio

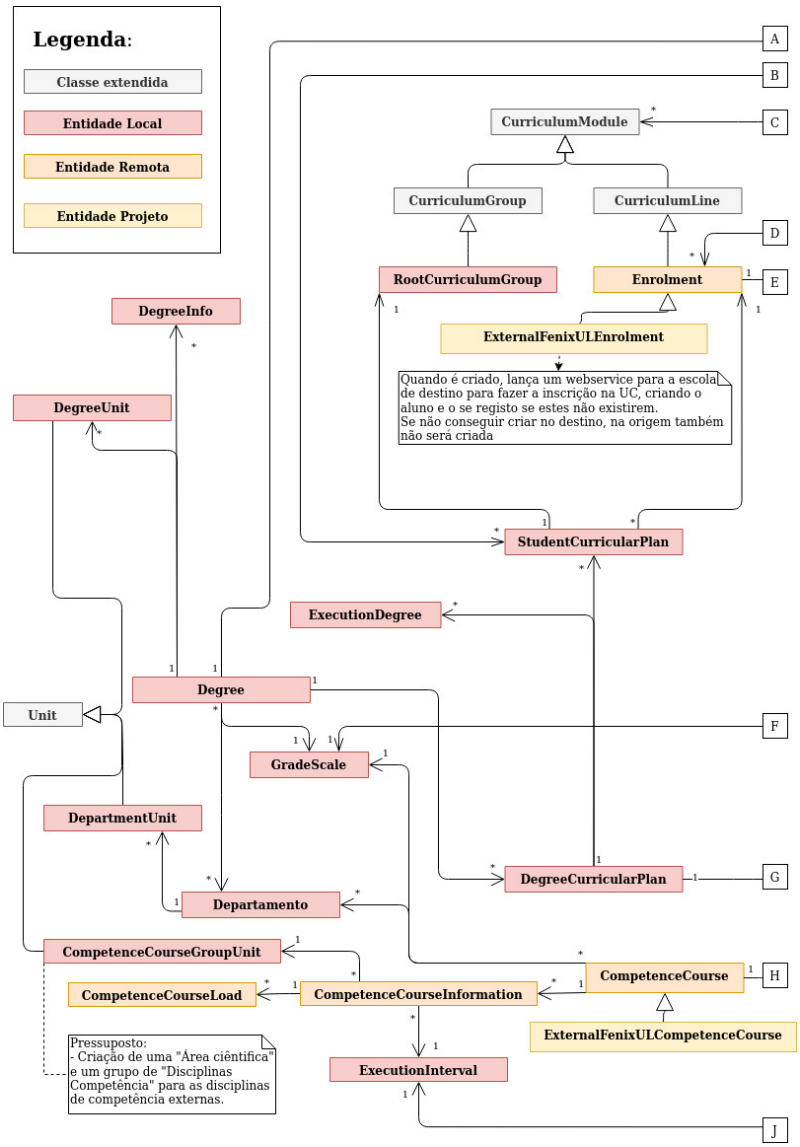

Figura A.1: Modelo de classes de domínio 1/3. As letras nas ligações do lado direito da imagem são referentes à página seguinte.

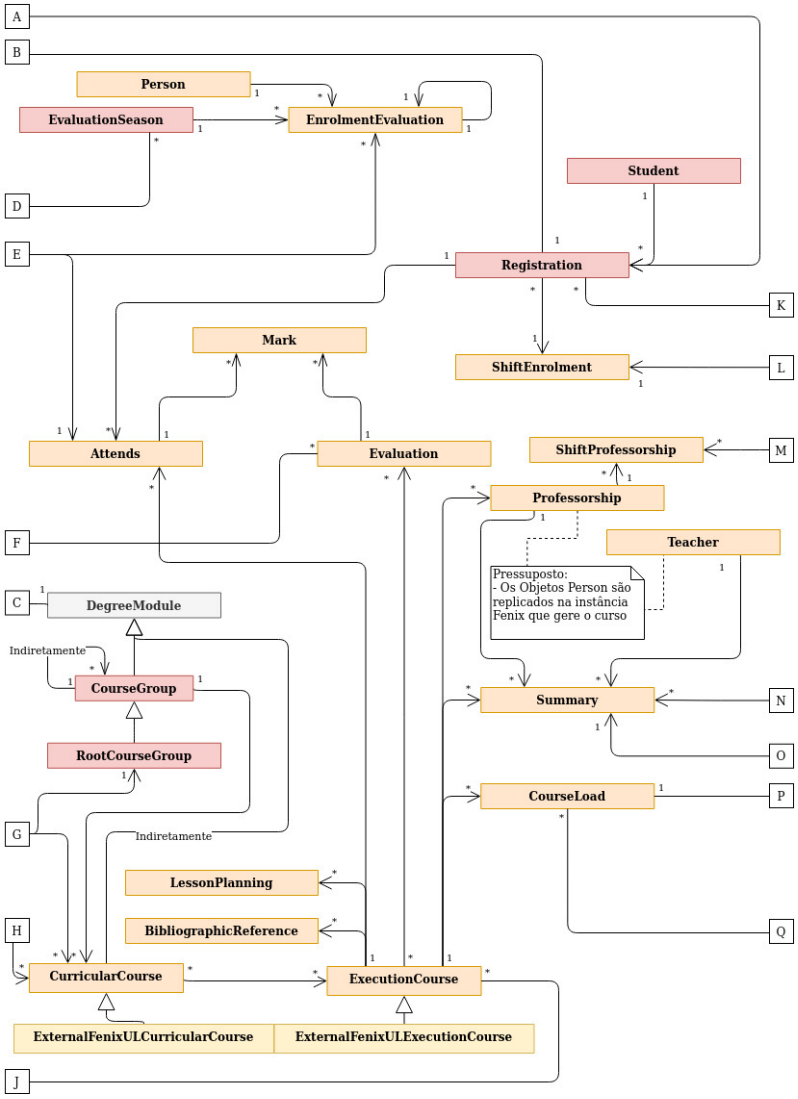

Figura A.2: Modelo de classes de domínio 2/3. As letras nas ligações do lado esquerdo da imagem são referentes à página anterior e do lado direito são referentes à página seguinte.

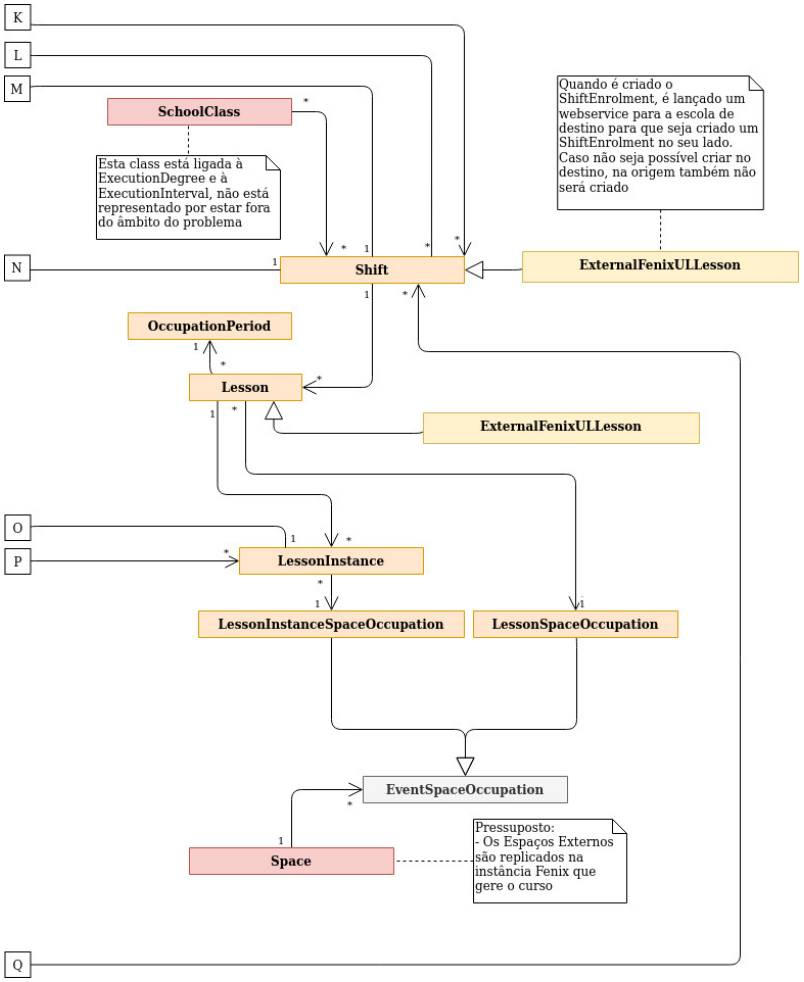

Figura A.3: Modelo de classes de domínio 3/3. As letras nas ligações do lado esquerdo da imagem são referentes à página anterior.